МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЛОСЬ Марія Василівна

Метод автоматичної генерації текстових сегментів із використанням BERT / Method for automated text segment generation using BERT

спеціальність: 122 - Комп'ютерні науки

освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконала студентка групи КНзм-21
М. В. Лось
_____

Науковий керівник:
к.т.н., доцент Х.В. Ліп'яніна Гончаренко
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
В.о. завідувача кафедри
_____ Н.В. Дзюбановська

ТЕРНОПІЛЬ - 2025

ЗМІСТ

ВСТУП

Стрімке зростання обсягів новинного контенту в онлайн-медіа, інформаційних агентствах та соціальних мережах зумовлює необхідність автоматизованої обробки текстів для підтримки аналітики, інформаційного моніторингу та персоналізованої доставки новин. Людина фізично не здатна опрацювати всі доступні публікації, тому ключову роль відіграють інтелектуальні системи, які здатні виділяти суттєву інформацію, формувати стислий виклад подій і подавати його у зручному для сприйняття вигляді. Центральним елементом таких систем є генерація коротких текстових сегментів – насамперед заголовків і підзаголовків, які концентрують основний зміст новини та визначають, чи зверне користувач увагу на матеріал.

У сучасній журналістиці заголовок виконує не лише інформаційну, а й атрактивну, навігаційну та прагматичну функції: він має бути лаконічним, змістовно насиченим, відповідати редакційним стандартам і водночас не спотворювати зміст новини. Для багатоджерельних стрічок новин, де поєднуються матеріали різних медіа з відмінною стилістикою, ручна підготовка заголовків є трудомісткою й економічно невигідною. Тому розробка методів автоматичної генерації текстових сегментів, здатних адаптуватися до різних тематик і стилів, набуває особливої значущості як для новинних порталів та агрегаторів, так і для аналітичних систем моніторингу.

Поява трансформерних мовних моделей, зокрема BERT (Bidirectional Encoder Representations from Transformers), суттєво змінила підходи до обробки природної мови. На відміну від класичних статистичних чи рекурентних моделей, BERT дає змогу будувати глибокі контекстуальні подання тексту, ураховуючи двонапрямлений контекст кожного токена. Це створює передумови для підвищення якості як екстрактивного, так і абстрактивного реферування, а також генерації заголовків на основі повних текстів статей. Однак інтеграція BERT-орієнтованих підходів у прикладний конвеєр обробки новин (від формування корпусу до оцінювання якості згенерованих сегментів) потребує формалізованого методу, адаптованого до специфіки новинного домену.

Актуальність теми магістерської кваліфікаційної роботи «Метод автоматичної генерації текстових сегментів із використанням BERT» зумовлена поєднанням кількох чинників: (i) зростаючою потребою у системах автоматизованого реферування та генерації заголовків для новинних ресурсів; (ii) необхідністю адаптації сучасних трансформерних моделей до реальних, часто «шумних» новинних корпусів; (iii) важливістю забезпечення балансу між інформативністю, стислістю та стилістичною відповідністю згенерованих сегментів редакційним вимогам. Додатковим аргументом на користь актуальності є можливість подальшого використання розробленого методу у прикладних задачах інформаційного моніторингу, аналізу медіапростору та інтелектуальних систем підтримки прийняття рішень.

Метою магістерської кваліфікаційної роботи є розробка та дослідження BERT-орієнтованого методу автоматичної генерації текстових сегментів для новинних текстів на основі публічного корпусу новин, з подальшим програмним впровадженням і кількісним оцінюванням якості результатів.

Для досягнення поставленої мети у роботі необхідно розв’язати такі основні завдання:

1) проаналізувати поняття текстових сегментів, особливості побудови заголовків новинних текстів та сформулювати вимоги до їх структури й змісту;

2) виконати аналітичний огляд методів екстрактивного та абстрактивного реферування текстів, а також підходів до автоматичної генерації заголовків;

3) дослідити трансформерні архітектури та модель BERT у задачах обробки природної мови, визначити їх придатність для генерації новинних заголовків;

4) сформувати корпус новинних текстів на основі публічного набору даних, здійснити його очищення, фільтрацію та первинну статистичну характеристику;

5) розробити BERT-орієнтований конвеєр попередньої обробки тексту, включаючи нормалізацію, токенізацію та формування вхідних тензорів;

6) спроєктувати архітектуру моделі генерації текстових сегментів на основі BertSummarizer або аналогічної BERT-орієнтованої модифікації;

7) реалізувати запропонований метод у середовищі Python, налаштувати процедуру навчання та тестування моделі на сформованих вибірках;

8) обрати та застосувати адекватні метрики оцінювання якості (наприклад, ROUGE-показники), порівняти результати із базовими підходами й проаналізувати сильні та слабкі сторони запропонованого рішення.

Об’єктом дослідження є процес автоматичної генерації коротких текстових сегментів (заголовків та підзаголовків) для новинних документів на основі трансформерних мовних моделей.

Предметом дослідження є методи, моделі та алгоритми автоматичної генерації текстових сегментів новинних статей із використанням BERT та його модифікацій, а також параметри попередньої обробки корпусу новинних текстів, що впливають на якість генерованих заголовків.

Для розв’язання поставлених завдань у роботі використовуються такі методи дослідження: аналіз і синтез наукових джерел у галузі обробки природної мови та нейронних мовних моделей; методи математичної статистики для аналізу корпусу новинних текстів; методи машинного навчання та глибокого навчання для побудови й навчання BERT-орієнтованої моделі; експериментальні методи для порівняльного оцінювання якості згенерованих текстових сегментів за стандартними метриками; програмна реалізація в середовищі Python із використанням спеціалізованих бібліотек для трансформерних моделей.

Наукова новизна одержаних результатів полягає в удосконаленні підходів до автоматичної генерації новинних текстових сегментів на основі BERT за рахунок: формалізації вимог до заголовків у мультиджерельному корпусі новин; побудови BERT-орієнтованого конвеєра попередньої обробки даних, інтегрованого з моделлю типу BertSummarizer; розроблення методу, що поєднує контекстуальні подання BERT із вимогами до довжини, інформативності та стилістичної відповідності заголовків. У межах роботи уточнено параметри

налаштування моделі на реальному корпусі новин та запропоновано рекомендації щодо підвищення якості генерації заголовків.

Практичне значення одержаних результатів полягає у створенні працездатного програмного модуля автоматичної генерації текстових сегментів новинних статей, реалізованого мовою Python із використанням BERT-орієнтованої архітектури. Розроблений модуль може бути інтегрований у системи новинних агрегаторів, платформи інформаційного моніторингу та аналітики медіаконтенту для формування заголовків і коротких анотацій. Окрім того, результати роботи доцільно використовувати в навчальному процесі при викладанні дисциплін, пов’язаних з інтелектуальним аналізом даних та обробкою природної мови, як приклад побудови прикладного модуля на основі сучасних трансформерних моделей.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження доповідалися й обговорювалися на студентської науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR-2025), яка відбулася в місті Тернополі 27–29 травня 2025 року та IX Міжнародної студентської наукової конференції «Модернізація та сучасні українські і світові наукові дослідження» 14 листопада 2025 в місті Житомир, Україна.

1 АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ТЕКСТОВИХ СЕГМЕНТІВ

1.1. Поняття текстових сегментів та задача генерації заголовків новинних текстів

Автоматична обробка новинних текстів базується на поданні документа як сукупності текстових сегментів – від окремих речень до більших смислових блоків, які описують певну подію, підподію або аспект теми. У класичних оглядах з автоматичного реферування тексту сегмент розглядається як елементарна одиниця відбору змісту (найчастіше речення), для якої обчислюються різні ознаки важливості та зв'язності [1–4]. У новинній журналістиці такі сегменти природно відповідають структурі статті: «лід» (короткий виклад події), основний текст, бекграунд і контекст.

Проблема сегментації тексту історично розвивалася як задача пошуку меж між підтемами. Один із класичних підходів – алгоритм TextTiling, який визначає кордони між багатопараграфними «плитками» на основі зміни лексичної когезії між сусідніми фрагментами [5]. У сучасних методах замість сирих частот слів часто використовують векторні подання речень чи абзаців, зокрема на основі попередньо натренованих мовних моделей, що дозволяє гнучкіше відображати семантичну близькість між сегментами [3, 4]. Для задачі генерації заголовків така сегментація дозволяє виділити найбільш інформативні частини статті, які мають бути відображені у короткій текстовій формулі заголовка.

Текстові сегменти можуть мати різну гранулярність: окремі речення, кванти інформації всередині речення (клауза, синтаксична група) або агреговані блоки з кількох речень, які утворюють мікротему. У новинному домені найбільш поширеним є реченнєвий рівень, оскільки саме речення зазвичай містить завершену інформаційну одиницю (факт, заяву, цитату) [1, 4]. Разом з тим, для побудови компактних заголовків часто виконують подальшу внутрішньо-реченнєву компресію – виділення ядра пропозиції та видалення другорядних обставин.

Новинні тексти, особливо у міжнародних корпусах (CNN/DailyMail, Gigaword тощо), характеризуються відносно стандартизованою риторичною структурою, де перший абзац концентрує ключову інформацію про те, «що сталося, де і коли» [9–11]. Це робить їх особливо придатними для задачі генерації заголовків: моделі можуть навчатися зіставляти повний текст статті з її коротким назвою, що розглядається як екстремально стиснутий текстовий сегмент.

Заголовок новинної статті виконує декілька функцій: інформаційну (конденсований опис ключової події), атрактивну (привернення уваги користувача), навігаційну (індексація у стрічці новин, пошуку, соцмережах) та прагматичну (задання тональності й фрейму інтерпретації) [6–8]. Це накладає жорсткі обмеження на довжину, стилістичні шаблони (відсутність дієслів-зв'язок, використання теперішнього часу, акцент на дію чи наслідок) та семантичну насиченість. У термінах автоматичної генерації заголовок можна розглядати як спеціальний тип текстового сегмента, що максимізує інформаційну щільність.

Формально задача генерації заголовків задається як умовна мовна модель: для вхідного документа (D) необхідно згенерувати послідовність токенів (y), яка максимізує ймовірність (P(θ)), де (θ) – параметри моделі [9–11]. Традиційні підходи будують цю модель у вигляді послідовнісних нейромереж (encoder–decoder) або трансформерів, що навчаються на великих паралельних корпусах «стаття–заголовок». Такі корпуси існують як для англомовних, так і для багатьох інших мов, що дозволяє переносити методи на мультимовні сценарії [6, 8].

У більш ранніх роботах заголовки часто будувалися екстрактивно: вибиралися речення з найбільшою вагомістю, наприклад за TF–IDF-оцінкою, позиційною евристикою чи графовими мірами центральності (PageRank/LexRank), після чого виконувалася ручна чи шаблонна компресія [1–3]. Поява нейронних моделей дозволила перейти до абстрактивного підходу, коли заголовок не обмежується підрядком вхідного тексту, а генерується заново,

що набагато краще відображає стилістичні норми журналістики, але водночас підвищує ризик «галюцинацій» – фабрикації фактів.

Сучасні дослідження зосереджуються на комбінуванні сегментації й генерації: модель спершу ідентифікує ключові сегменти статті, а потім використовує їх як стиснутий контекст для побудови заголовка [6–8]. При цьому можуть використовуватись додаткові сигнали – інформація про рубрику, джерело, час публікації – які дозволяють уточнити фокус заголовка (наприклад, акцент на політичних наслідках, економічних показниках, гуманітарному вимірі тощо).

Окремий напрямок досліджень – персоналізована генерація заголовків, коли текстові сегменти адаптуються до інтересів конкретного користувача або групи аудиторії (наприклад, експертна vs масова аудиторія). Такі моделі комбінують контентні ознаки статті з профілем користувача, але водночас підсилюють ризики маніпулятивних чи клікбейтних заголовків, що ставить питання етичних та регуляторних обмежень [3, 4].

У рамках даної роботи текстові сегменти розглядаються як компактні, семантично завершені фрагменти новинного тексту, придатні для використання як заголовки або підзаголовки. Використання BERT-подібних моделей дозволяє отримувати контекстуально-залежні подання цих сегментів і навчати систему, яка автоматично відбирає або генерує найбільш інформативні та стилістично адекватні заголовки, інтегровані в загальний конвеєр обробки новинного контенту [16–18].

1.2. Методи екстрактивного та абстрактивного реферування текстів

Автоматичне реферування текстів традиційно поділяють на два основні підходи – екстрактивний та абстрактивний, які відрізняються тим, як формується підсумковий текст. В оглядах Ненкової та МакКіон [1], Аллагярі та співавт. [3] і більш нових систематичних дослідженнях [4] підкреслюється, що екстрактивні методи здійснюють відбір наявних речень або фраз із документа,

тоді як абстрактивні – генерують нові формулювання на рівні значень, подібно до того, як це робить людина.

Екстрактивне реферування можна розглядати як задачу ранжування або двокласової класифікації речень: кожному реченню ставиться у відповідність оцінка «важливості», після чого обирається підмножина речень, яка максимізує сумарну інформативність і мінімізує надмірність [1, 2]. Класичні підходи використовують статистичні ознаки (TF–IDF, позиція в документі, наявність ключових слів), графові моделі (TextRank, LexRank) та оптимізаційні формулювання (цілочисельне програмування) для глобального вибору підмножини речень.

У подальших роботах екстрактивні моделі стали поєднуватися з машинним навчанням: ознаки речень подаються на вхід класифікатору (SVM, логістична регресія, нейромережі), який вчиться розрізняти «резюме-дружні» речення на основі анотацій людини [1, 3]. З появою попередньо натренованих мовних моделей (BERT, RoBERTa тощо) екстрактивні методи отримали доступ до глибоких контекстуальних ознак: модель може враховувати не лише локальну лексичну інформацію, а й глобальний дискурсивний контекст [17, 18].

Абстрактивне реферування значно складніше, оскільки вимагає повноцінної генерації природної мови, включно з перефразуванням, агрегацією та компресією інформації [4]. Ранні системи спиралися на шаблонні та лінгвістично орієнтовані методи (глибинний синтаксичний аналіз, граматичні шаблони), які мали обмежену гнучкість. Прорив стався з появою нейронних послідовнісних моделей (seq2seq) з механізмом уваги, які інтерпретують реферування як задачу машинного перекладу з «мови документа» на «мову реферату» [9, 11].

Модель Rush–Chopra–Weston продемонструвала, що нейронна модель з увагою здатна генерувати якісні заголовки й короткі резюме на корпусі Gigaword, хоч і з певними обмеженнями щодо довжини та точності фактів [9]. Подальші роботи Налаппаті та співавт. розширили ці підходи на багатореченнєві резюме [11]. Важливим кроком став механізм pointer-generator, що комбінує можливість копіювати слова з джерела з генерацією нових токенів,

тим самим поєднуючи сильні сторони екстрактивного та абстрактивного підходів [10].

Для новинних текстів pointer-generator-мережі суттєво зменшують кількість фактичних помилок та повторів, зберігаючи водночас здатність до перефразування [10]. На основі цих ідей побудовано багато модифікацій, включно із спеціалізованими моделями для генерації заголовків, де важливо не лише стисло передати зміст, а й дотримуватися характерних стилістичних шаблонів заголовків [6–8].

Сучасні огляди показують, що межа між екстрактивним та абстрактивним реферуванням дедалі більше розмивається: моделі із попереднім навчанням можуть реалізовувати гібридні підходи, де екстракція використовується як проміжний крок (виділення ключових сегментів), а абстракція – для генерації компактного, стилістично гладкого тексту [3, 4, 18]. У практичних системах це особливо важливо для контролю над фактичною коректністю результатів.

Окрема проблема – оцінювання якості рефератів. Найпоширенішим є метричне сімейство ROUGE, яке порівнює n-грамну, послідовну та парну подібність між автоматично згенерованими й еталонними рефератами [12]. Хоча ROUGE добре корелює з людськими оцінками на новинних корпусах, воно погано враховує перефразування та семантичні перетини. Частково тому в оглядах рекомендують комбінувати його з іншими метриками, зокрема BLEU, та, по можливості, людським оцінюванням [4, 13].

У новинному домені сформувалися стандартні корпуси для навчання й тестування моделей реферування (CNN/DailyMail, XSum, Gigaword тощо), які містять як повні статті, так і резюме чи заголовки, створені редакторами [9–11, 18]. Це дозволяє систематично порівнювати різні архітектури та стратегії навчання.

Для задачі генерації текстових сегментів у вигляді заголовків особливе значення має екстремальний рівень компресії, коли результат містить лише кілька слів. Тут стандартні методи екстрактивного реферування зазвичай недостатні, і перевагу отримують абстрактивні моделі, доповнені механізмами копіювання й структурної уваги [6–8, 10]. Екстрактивні компоненти можуть

використовуватися для відбору найбільш релевантних сегментів, на основі яких будується компактний заголовок.

У контексті цієї роботи доцільно розглядати екстрактивні підходи як базову лінію (baseline), яка гарантує збереження фактів, а абстрактивні – як шлях до покращення читабельності й стилістичної природності сгенерованих текстових сегментів. Поєднання обох підходів у рамках BERT-орієнтованої архітектури дозволяє будувати модуль, здатний формувати новинні заголовки з високою інформативністю, компактністю та відповідністю редакційним стандартам [17, 18].

1.3. Трансформерні архітектури та модель BERT у задачах обробки природної мови

Впровадження трансформерних архітектур стало переломним моментом в обробці природної мови, дозволивши відмовитися від рекурентних та згорткових структур на користь повністю уваго-орієнтованих моделей [14]. Модель Transformer, запропонована Васвані та співавт., показала, що багатоголова самоувага в поєднанні з позиційним кодуванням і блоками прямого поширення (feed-forward) здатна ефективно моделювати довгострокові залежності в послідовностях, забезпечуючи при цьому високу паралелізацію навчання [14].

Архітектурно Transformer складається з стеку однакових шарів, кожен з яких містить підшар самоуваги та підшар позиційно-незалежної нелінійної трансформації, оточені резидуальними зв’язками та операціями нормалізації [14]. У self-attention кожне слово (токен) взаємодіє з іншими через механізм «ключ–запит–значення», що дозволяє моделі динамічно перерозподіляти вагу між різними частинами контексту. Багатоголовість (multi-head) розщеплює цей процес на кілька «голів», кожна з яких фокусується на власних аспектах структури й значення.

Ідеї самоуваги розвивалися і до Transformer – зокрема, у роботі Lin та співавт. запропоновано структуроване самоуважне подання речення, де

матричне кодування дозволяє різним «рядам» уваги зосереджуватися на різних семантичних компонентах висловлювання [15]. Ці підходи продемонстрували, що увага може слугувати не лише допоміжним механізмом, а й центральним будівельним блоком моделі.

На базі трансформерної архітектури була розроблена модель BERT (Bidirectional Encoder Representations from Transformers), яка запровадила парадигму глибокого двонапрямленого попереднього навчання мовних моделей [16]. На відміну від однобічних чи авто-регресивних підходів, BERT одночасно враховує як лівий, так і правий контекст кожного токена, що дозволяє формувати більш багаті семантичні подання. Попереднє навчання здійснюється на завданнях маскованого моделювання мови (MLM) та передбачення наступного речення (NSP), що дає змогу моделі опанувати як локальні, так і міжреченнєві зв’язки.

BERT продемонстрував суттєві покращення в широкому спектрі задач – від класифікації текстів та аналізу настроїв до розпізнавання іменованих сутностей і відповіді на запитання (SQuAD, GLUE тощо) [16]. Ключовою особливістю є схема «fine-tuning»: замість того, щоб навчати модель з нуля для кожної задачі, BERT попередньо тренується на великих некерованих корпусах, після чого до нього додається тонкий задачно-орієнтований шар, який адаптується на специфічних даних.

Задачі реферування та генерації заголовків стали природним полем застосування BERT, оскільки вони критично залежать від глибокого розуміння контексту документа. У роботі Liu (BERTSUM) було показано, що адаптація BERT до екстрактивного реферування – шляхом введення спеціальних токенів для меж речень та надбудови міжреченних трансформерних шарів – дозволяє досягти найкращих на той час результатів на корпусі CNN/DailyMail [17].

Подальша робота Liu та Lapata запропонувала загальну рамку для текстового реферування з попередньо натренованими енкодерами, яка об’єднує екстрактивні й абстрактивні моделі на основі BERT-подібних представлень [18]. Автори показали, що введення документного енкодера на основі BERT та додаткових міжреченних трансформерних шарів дозволяє ефективно

моделювати довгострокову дискурсивну структуру документа, а спеціальний двоетапний режим тонкого налаштування покращує якість абстрактивних резюме.

Окрім реферування, BERT активно використовується і в задачах текстової сегментації. Наприклад, Solbiati та співавт. продемонстрували, що BERT-ембеддинги можна застосувати для некерованої тематичної сегментації стенограм наради, досягаючи суттєвого зниження помилки порівняно з попередніми методами [19]. Це підтверджує придатність BERT для моделювання змін тематики й семантичних «зламів» у довгих текстах, що є критично важливим і для генерації узгоджених текстових сегментів.

Існують також варіанти BERT, оптимізовані для сегментації на рівні підслів (наприклад, BERTSeg для сегментації субслів у задачах машинного перекладу), де контекстуальні подання використовуються для вибору оптимальних місць поділу всередині слова [18, 20]. Хоча ці моделі орієнтовані на іншу гранулярність, їхня ідеологія – використання контекстуально збагачених ембеддингів для прийняття рішень про структуру послідовності – є релевантною й для сегментації на рівні речень та абзаців.

У контексті генерації заголовків і коротких текстових сегментів BERT зазвичай використовується як енкодер, який забезпечує високоякісне подання документа, тоді як декодер реалізується або окремою трансформерною мережею, або механізмами копіювання в рамках гібридних архітектур [6–8, 18]. Такі моделі можуть одночасно враховувати глобальну структуру тексту, локальні ключові факти й стилістичні особливості цільового регістру (новинний заголовок).

Ще одна важлива перевага трансформерних архітектур – масштабованість. Завдяки повній паралелізації обчислень по токенах вони добре працюють на сучасних GPU-кластерах, що дозволяє тренувати моделі з сотнями мільйонів або навіть мільярдами параметрів на багатомовних та мультидоменних корпусах [14, 16]. У результаті стає можливим переносити одну й ту ж архітектуру BERT-типу на різні мови (включно з українською) та різні жанри текстів, зберігаючи загальну структуру моделі.

Таким чином, трансформерні архітектури і, зокрема, модель BERT створюють потужну основу для задач автоматичної генерації текстових сегментів і заголовків новинних текстів. Вони дозволяють поєднувати глибоке контекстуальне розуміння документа (через self-attention) з гнучкими стратегіями екстрактивного й абстрактивного реферування, а також інтегрувати в одну модель як відбір сегментів, так і їх мовну реалізацію. Це робить BERT ключовим компонентом у побудові сучасних систем автоматичної генерації заголовків, що і закладається в основу подальших розділів роботи.

1.4 Постановка задачі

Актуальність дослідження зумовлена стрімким зростанням обсягів новинного контенту в цифровому середовищі. Сучасні медіаплатформи, агрегатори новин та соціальні мережі щоденно генерують тисячі публікацій, і саме заголовок стає ключовим навігаційним елементом, що визначає, чи зверне користувач увагу на матеріал. Заголовки виконують водночас інформаційну, атрактивну, навігаційну та прагматичну функції, задаючи інтерпретаційний фрейм події та впливаючи на сприйняття змісту [6–8]. В умовах інформаційного перевантаження проблема автоматизованого формування коротких, інформативних та стилістично коректних текстових сегментів (заголовків, підзаголовків, тизерів) для новинних текстів набуває особливого значення як для підвищення ефективності роботи редакцій, так і для покращення користувацького досвіду.

З наукової точки зору задача автоматичної генерації заголовків є частковим випадком проблеми текстового реферування, яка традиційно розвивається у двох напрямках – екстрактивному та абстрактивному [1–4]. Класичні екстрактивні методи, що базуються на TF–IDF-вагах, позиційних евристиках, графових моделях (TextRank, LexRank) або простих класифікаторах, добре зберігають фактичну інформацію, проте обмежені в здатності відтворювати характерні стилістичні конструкції новинних заголовків і досягати екстремальної компресії тексту [1–3, 10]. Нейронні seq2seq-моделі та

трансформерні декодери забезпечили перехід до абстрактивного реферування [9–11], однак проблема контрольованої генерації дуже коротких сегментів із дотриманням редакційних норм та мінімізацією «галюцинацій» залишається відкритою й активно досліджується в сучасній літературі [4, 6–8, 18].

Запровадження трансформерних архітектур і, зокрема, моделі BERT стало переломним етапом у розвитку методів обробки природної мови [14–16]. BERT-подібні енкодери забезпечують глибоке двонапрямлене моделювання контексту й демонструють найкращі на сьогодні результати в задачах класифікації текстів, аналізу тональності, відповіді на запитання та реферування [16–18]. Разом із тим, питання побудови спеціалізованих BERT-орієнтованих конвеєрів для генерації коротких новинних заголовків, які поєднують екстрактивні та абстрактивні стратегії, досі вивчене недостатньо. Актуальним є дослідження архітектур типу «BERT-енкодер – трансформерний декодер», здатних одночасно моделювати глобальну структуру документа і відтворювати жорсткі обмеження на довжину, інформативність та стилістичні патерни заголовків [17, 18].

Додатковою складовою актуальності є використання реалістичного мультиджерельного корпусу новин, сформованого на основі публічного набору даних News Headlines & Summary from select 12 sources з платформи Kaggle. Корпус містить понад 35 тис. унікальних заголовків та повні тексти англомовних новин з різних медіаресурсів, включно з Reuters та CBS News, доповнені часовими та системними атрибутами. Така структура забезпечує різноманіття тематик, редакційних стилів і політичних контекстів, що ускладнює задачу автоматичної генерації, але водночас робить результати дослідження релевантними для реальних сценаріїв обробки новинного потоку.

Практична значущість дослідження пов’язана з широким спектром застосувань автоматично згенерованих текстових сегментів у сучасних інформаційних системах. Якісний заголовок є ключовим елементом для ранжування записів у стрічці новин, побудови рекомендацій, формування push-сповіщень, а також для адаптації контенту під різні канали доставки (мобільні додатки, вебінтерфейси, соціальні мережі). Інтеграція BERT-орієнтованих моделей генерації сегментів у редакційні системи дозволяє зменшити

навантаження на журналістів, прискорити випуск матеріалів, підтримувати єдині стилістичні стандарти й водночас знижувати ризики маніпулятивних або клікбейтних формулювань завдяки чітко налаштованим обмеженням і процедурам оцінювання якості [3, 4, 6–8].

Отже, дослідження, присвячене розробці та практичній реалізації методу автоматичної генерації текстових сегментів новинних статей із використанням BERT, є актуальним як з теоретичної, так і з прикладної точки зору. З теоретичного боку воно спрямоване на уточнення й розвиток підходів до екстремально стислого абстрактивного реферування на основі трансформерних архітектур [17, 18]. З прикладної – дозволяє побудувати прототип інтелектуального модуля, що забезпечує автоматизоване формування заголовків на реальному мультиджерельному корпусі новин, із подальшою оцінкою якості за метриками BLEU та ROUGE, що наближає результати роботи до практичного впровадження в системах медіамоніторингу та новинних сервісах.

Метою магістерської кваліфікаційної роботи є розробка та дослідження BERT-орієнтованого методу автоматичної генерації текстових сегментів для новинних текстів на основі публічного корпусу новин, з подальшим програмним впровадженням і кількісним оцінюванням якості результатів.

Для досягнення поставленої мети у роботі необхідно розв’язати такі основні завдання:

1) проаналізувати поняття текстових сегментів, особливості побудови заголовків новинних текстів та сформулювати вимоги до їх структури й змісту;

2) виконати аналітичний огляд методів екстрактивного та абстрактивного реферування текстів, а також підходів до автоматичної генерації заголовків;

3) дослідити трансформерні архітектури та модель BERT у задачах обробки природної мови, визначити їх придатність для генерації новинних заголовків;

4) сформувати корпус новинних текстів на основі публічного набору даних, здійснити його очищення, фільтрацію та первинну статистичну характеристику;

5) розробити BERT-орієнтований конвеєр попередньої обробки тексту, включаючи нормалізацію, токенізацію та формування вхідних тензорів;

6) спроєктувати архітектуру моделі генерації текстових сегментів на основі BertSummarizer або аналогічної BERT-орієнтованої модифікації;

7) реалізувати запропонований метод у середовищі Python, налаштувати процедуру навчання та тестування моделі на сформованих вибірках;

8) обрати та застосувати адекватні метрики оцінювання якості (наприклад, ROUGE-показники), порівняти результати із базовими підходами й проаналізувати сильні та слабкі сторони запропонованого рішення.

Висновки до розділу 1

У першому розділі виконано систематизований аналітичний огляд підходів до автоматичної генерації текстових сегментів, у межах якого уточнено поняття текстового сегмента та специфіку заголовка як особливого типу стислого, семантично насиченого фрагмента новинного тексту. Показано, що заголовок поєднує інформаційну, атрактивну, навігаційну та прагматичну функції, а отже, задачі його автоматичної побудови не можна зводити лише до механічної компресії чи вибірки речень. Узагальнення існуючих досліджень засвідчує, що успішні моделі мають одночасно враховувати структурну організацію новинного тексту та редакційні вимоги до заголовків.

Проаналізовано два базові класи методів текстового реферування – екстрактивні та абстрактивні, а також їх гібридні варіанти. Показано, що екстрактивні підходи (на основі TF–IDF, графових моделей, класичних класифікаторів) добре зберігають фактичну інформацію, але обмежені щодо стилістичної гнучкості та здатності генерувати екстремально стислі заголовки. Абстрактивні нейронні моделі типу seq2seq з механізмами уваги та pointer-generator долають ці обмеження, проте породжують низку нових викликів, пов’язаних з контролем довжини, уникненням повторів і мінімізацією «галюцинацій» під час генерації.

Окрему увагу приділено трансформерним архітектурам і моделі BERT як основі сучасних систем обробки природної мови. Показано, що BERT-подібні енкодери забезпечують глибоке двонапрямлене моделювання контексту, що є критично важливим для задач реферування та побудови заголовків, а також можуть бути інтегровані в екстрактивні, абстрактивні та гібридні схеми. За результатами огляду сформульовано постановку задачі й обґрунтовано доцільність розроблення BERT-орієнтованого методу автоматичної генерації новинних текстових сегментів, що визначило зміст наступних розділів роботи.

2 РОЗРОБКА МЕТОДУ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ТЕКСТОВИХ СЕГМЕНТІВ

2.1. Характеристика корпусу новинних текстів та вимоги до заголовків

Корпус новинних текстів сформовано на основі публічного набору даних *News Headlines & Summary from select 12 sources* з платформи Kaggle [20], який містить заголовки, описи та повні тексти новинних статей за останні місяці. У вихідному файлі full_data.csv для кожної статті збережено метадані: джерело (source), автор (author), заголовок (title), короткий опис (description), URL-адресу першоджерела (url), дату запиту до API (requested_date), дату публікації (publishedAt) та повний текст новини (content). Структуру набору даних і приклади записів проілюстровано на рисунку 2.1, де видно, що корпус поєднує як текстові, так і часові та системні атрибути.

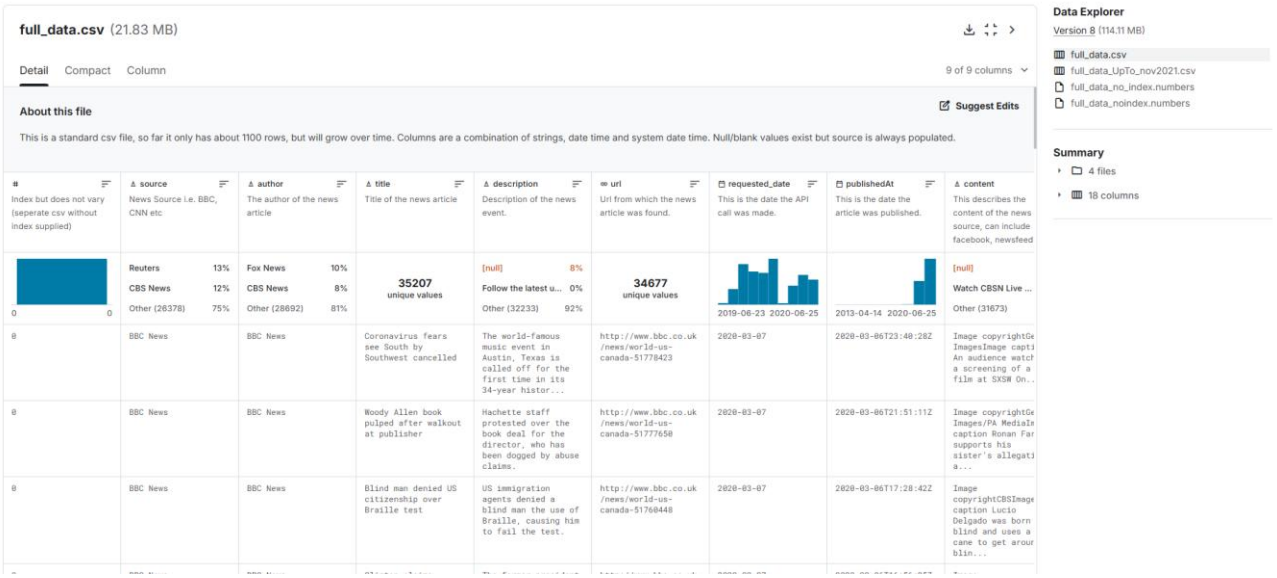

Рисунок 2.1 – Структура корпусу новинних текстів у наборі даних *News Headlines & Summary from select 12 sources* [20]

Аналіз розподілу значень у стовпці source показує, що корпус є мультиджерельним: переважають матеріали агентства Reuters (приблизно 13 % від загальної кількості записів) і телеканалу CBS News (близько 12 %), тоді як решта понад 70 % статей належить до категорії «Other», яка агрегує інші 10+

медіа-ресурсів. Така структура (див. гістограму частот на рис. 2.1 та табл.2.1) забезпечує різноманіття редакційних стилів і політичних упереджень, що є важливим для узагальнювальної моделі заголовків.

Таблицю 2.1 – Структура корпусу новинних текстів за джерелами, авторами та основними полями [20]

Показник	Категорія / поле	Значення	Коментар
Джерело новин (source)	Reuters	13 %	Один із двох основних постачальників новин у корпусі.
	CBS News	12 %	Друге за часткою джерело новин.
	Інші джерела	75 %	Сукупність решти 10 джерел; домінуюча частка корпусу.
Автор (author)	Fox News	10 %	Значна частка матеріалів належить каналу Fox News.
	CBS News	8 %	Помітна, але менша частка записів, ніж у Fox News.
	Інші автори / канали	81 %	Висока фрагментованість за авторами; переважають поодинокі автори.
Структура полів	title (заголовок)	35 207 унікальних значень	Майже кожна новина має унікальний заголовок; висока варіативність.
	description (опис)	32 233 унікальних значень	≈92 % унікальних описів; близько 8 % записів мають пропуск (null).
	url (посилання на джерело новини)	34 677 унікальних значень	Майже 1:1 відповідність між записом у корпусі та URL-адресою новини.

Стовпець author має подібну картину: частину записів становлять агреговані автори на кшталт *Fox News* чи *CBS News*, але понад 80 % значень належить до категорії «Other», де згруповано індивідуальних журналістів або матеріали без чітко вказаного автора. Це означає, що у корпусі часто відсутній стабільний авторський стиль, а модель має навчатися на змішаній стилістиці, характерній для новинних редакцій, а не окремих письменників.

Ключову роль для задачі генерації заголовків відіграють поля title та content. У наборі даних зафіксовано понад 35 000 унікальних заголовків, що

свідчить про високу варіативність формулювань навіть за наявності повторюваних тематик (політика США, пандемія COVID-19, економічні новини тощо). Поле content містить повні тексти новин, однак значна частка рядків описана агрегованою категорією «Other», що вказує на неповну доступність повних текстів для частини записів. Це зумовлює потребу очищення даних, видалення порожніх записів і нормалізації символів, що було реалізовано в попередніх етапах препроцесингу.

Часові поля requested_date та publishedAt забезпечують можливість аналізу еволюції тем у часі. Згідно з оглядом у середовищі Kaggle (див. рис. 2.1), основна маса статей охоплює період приблизно шести місяців, причому спостерігається помітна концентрація публікацій у періоди активізації інформаційних приводів (спалахи COVID-19, виборчі кампанії, масові протести). Для задачі генерації заголовків це означає високу тематичну коваріацію в межах короткого інтервалу часу, що збільшує складність моделювання.

2.2 BERT-орієнтована попередня обробка тексту

BERT-орієнтована попередня обробка тексту розглядається як формальний етап перетворення сирого тексту T у набір числових тензорів, сумісних з архітектурою трансформера: індекси токенів, позиційні індекси, сегментні мітки та маски уваги. На відміну від «класичної» попередньої обробки, її ціль не стільки в агресивному фільтруванні тексту, скільки в збереженні максимальної кількості інформації при переведенні її у дискретний словник субслів, на основі якого BERT попередньо навчався. У цьому сенсі попередня обробка виступає як детермінована функція

$$f: RawText \rightarrow (token_{ids}, segment_{ids}, position_{ids}, attention_{mask}). \quad (2.1)$$

На першому рівні здійснюється нормалізація символів: вирівнювання регістру (для моделей типу bert-base-uncased — перетворення всіх букв до нижнього регістру), уніфікація пробілів, видалення керівних символів та

артефактів форматування (переведення рядка, зайві слеші тощо). Формально, для вхідного рядка T застосовується композиція перетворень

$$T' = g_k\left(g_2\left(g_{1(T)}\right)\right), \tag{2.2}$$

де кожна g_i — елементарна операція (наприклад, заміна $\backslash n \rightarrow$ пробіл). Ці кроки мінімізують шум, не змінюючи лексичну структуру тексту.

Далі виконується базова сегментація на речення та токени слів, як правило за допомогою лінгвістичного пайплайна (наприклад, spaCy або аналогічних інструментів). Цей крок задає послідовність лексичних одиниць $w = (w^1, \ldots, w_N)$ до того, як буде застосована субсловникова токенізація. Важливо, що на цьому етапі небажано радикально видаляти пунктуацію та спеціальні символи, оскільки BERT навчався саме на текстах з реальною пунктуацією, а частина таких символів може бути важливим стилістичним маркером.

Ключовим елементом BERT-орієнтованої попередньої обробки є субсловникова токенізація WordPiece, що відображає кожне слово w_i в послідовність субслів $s_{i,1}, \ldots, s_{i,}k_i$ з фіксованого словника розміру V ($\approx$ 30 000 для $bert - base$). Формально маємо відображення

$$\varphi: w \rightarrow (s^1, \ldots, s_m), \text{де } s_j \in V \text{ і } m \leq m_{max} \tag{2.3}$$

Часто використовується префіксальна нотація (наприклад, ##ing), що явно кодує належність субслова до середини чи кінця слова. Це дозволяє одночасно: (i) не роздувати словник за рахунок рідкісних форм; (ii) зберігати морфологічну інформацію через спільні префікси/суфікси.

Після субсловникової токенізації формується послідовність токенів моделі $t = (t^1, \ldots, t_n), \text{де } n \leq L_{\max\,(L_{\max\,\text{зазвичай}\,512})}$. До неї додаються спеціальні маркери [CLS] на початку послідовності та [SEP] для розділення сегментів (наприклад, «[CLS] текст запиту [SEP] текст документа [SEP]»). Математично це відображення $\psi: (s^1, \ldots, s_m) \rightarrow \left(t^0, t^1, \ldots, t_m, t_{\{m+1\}}\right)$, де $t^0 = [CLS], t_{\{m+1\}} = [SEP]$. Для задач з двома реченнями або «запит–контекст» формується конкатенація двох послідовностей із внутрішнім [SEP].

Наступний крок — перетворення токенів у індекси словника. Для кожного токена t_i визначається ціле число $x_i \in \{0, \dots, V-1\}$, і вхідний текст подається у вигляді вектора індексів $x = (x^1, \dots, x_n)$. На рівні ембеддингів кожен індекс x_i відображається у вектор $e_{tok(x_i)} \in \mathbb{R}^d$, де d — розмірність ембеддингу (d = 768 для bert-base). Таким чином, токенізація + ембеддинг реалізують відображення

$$x_i \rightarrow e_{tok(x_i)}, i = 1, \dots, n, \tag{2.4}$$

яке переводить дискретні символи у щільний векторний простір.

Для повної вхідної репрезентації використовуються адитивні ембеддинги токена, позиції та сегмента. Якщо p_i — позиційний індекс $(0, \dots, L_{max}-1)$ s_i — сегментний індекс (0 або 1 для двох речень), то кінцеве подання і-го елемента послідовності задається як

$$h_i = e_{tok(x_i)} + e_{pos(p_i)} + e_{seg(s_i)}. \tag{2.5}$$

У матричній формі отримуємо $H \in \mathbb{R}^{\{n \times d\}}$, де кожен рядок $H_i = h_i$. Саме Н подається на вхід багатоголовій механіці уваги трансформера, що і забезпечує контекстуалізацію субслів.

Паралельно формується маска уваги $M \in \{0,1\}^n$, яка визначає, які позиції є «реальними» токенами, а які — паддингом. Елемент $M_i = 1$, якщо позиція і належить до справжнього тексту, та $M_i = 0$, якщо це заповнювач. При побудові батчів послідовності вирівнюються до фіксованої довжини L_{batch}, і маска гарантує, що увага не розповсюджується на штучно додані паддингові позиції. Математично це реалізується у вигляді додавання великого негативного зсуву до логітів уваги для токенів з $M_i = 0$.

У задачах на кшталт автоматичної генерації заголовків, де використовується BERT-орієнтований енкодер і окремий декодер, попередня обробка розділяється на вхідну та цільову гілки. Вхідний текст обробляється, як описано вище, а для цільових послідовностей (наприклад, заголовків) будується окремий субсловниковий словник та додаються маркери початку та кінця послідовності (наприклад, $< s >$). Формально для цільової послідовності $y = (y^1, \dots, y_T)$ будується розширена послідовність $\left(y^0 = < s >, y^1, \dots, y_T, y_{\{T+1\}}\right)$ з власними індексами та ембеддингами, яку споживає декодер.

Особливе значення в BERT-орієнтованій попередній обробці має обробка довгих документів, довжина яких перевищує L_{max}иповим підходом є застосування ковзних вікон із перекриттям: документ розбивається на підпослідовності довжини L_{max}перекриттям k токенів, тобто формується множина вікон $W_j = \left(t_{\{j\cdot(L_{max}-k)+1\}}, \ldots, t_{\{j\cdot(L_{max}-k)+L_{max}\}}\right)$оляє зберегти локальний контекст у кожному вікні, при цьому дотримуючись апаратних обмежень моделі. Візуальні схеми таких пайплайнів (нормалізація → токенізація → побудова ембеддингів → маски уваги) можна знайти, наприклад, у типовій діаграмі «Overview of text preprocessing, tokenization stages, and BERT model pretraining steps».

Таким чином, BERT-орієнтована попередня обробка задає строгу послідовність перетворень від сирого тексту до набору тензорів (H, M, s, p), узгоджених із словником і навчальними гіперпараметрами конкретної BERT-моделі. У більш формальному вигляді це можна записати як композицію

$$f = f_{mask} \times f_{embed} \times f_{spec} \times f_{wp} \times f_{norm}, \quad (2.6)$$

де f_{norm} — нормалізація тексту, f_{wp} — WordPiece-токенізація, f_{spec} — додавання спеціальних токенів і сегментів, f_{embed} — складання ембеддингів, f_{mask} — побудова масок. Від якості та стабільності цієї композиції безпосередньо залежить здатність моделі коректно відтворювати контекст і, відповідно, генерувати адекватні текстові сегменти.

2.3 Архітектура моделі генерації текстових сегментів на основі BertSummarizer

Архітектура моделі генерації текстових сегментів на основі BertSummarizer може бути формалізована як функція

$$f(\theta): X \rightarrow Y, \quad (2.7)$$

де X — простір вхідних текстів (новинні статті), а Y — простір цільових коротких текстових сегментів (заголовків). У термінах послідовностей це відображення

$$(x^1, \ldots, x_n) \mapsto (y^1, \ldots, y_T), \quad (2.8)$$

де $(x^1, \ldots, x_n)$ — токени вхідного документа, а $(y^1, \ldots, y_T)$ — токени згенерованого заголовка фіксованої або змінної довжини $T \leq 50$. Модель реалізує це відображення в парадигмі encoder–decoder з використанням попередньо навченого BERT як енкодера контексту.

На рівні енкодера попередньо навчена модель bert-base-uncased відображає послідовність токенів $x = (x^1, \ldots, x_n)$ у матрицю контекстних представлень

$$H = Enc_{BERT(x)} \in R^{\{n \times d\}}, \tag{2.9}$$

де $d = 768$ — розмірність ембеддингу, а кожен рядок $h_i \in R^{\{d\}}$ є бідирекційним контекстуальним вектором для токена x_i. Таким чином, BERT забезпечує щільну, семантично збагачену репрезентацію вхідного тексту, яка виступає фіксованим «контекстним простором» для подальшої генерації заголовка декодером.

Декодер у BertSummarizer будується як стек з $L_{dec} = 4$ трансформерних шарів, організованих за класичною схемою encoder–decoder: кожен шар містить (1) блок самоуваги над уже згенерованою частиною вихідної послідовності, (2) блок міжпослідовної (cross-) уваги до матриці H енкодера та (3) двошаровий feed-forward модуль. На формальному рівні, якщо позначити матрицю прихованих станів декодера на k-му рівні як $D^{\{(k)\}} \in R^{\{T \times d\}}$, то кожен шар реалізує композицію перетворень

$$D^{\{(k)\}} = FFN\left(MHA_{dec}\left(D^{\{(k-1)\}}, H \right)\right) \tag{2.10}$$

з додаванням резидуальних зв'язків і нормалізації за шаром, як у вихідному трансформері.

Ключовим елементом є механізм багатоголової уваги (multi-head attention) з $h = 2$ головами. Для кожної голови h_j обчислюються матриці запитів Q_j, ключів K_j та значень V_j шляхом лінійних перетворень $D^{\{(k-1)\}}$ або H. Одна голова реалізує

$$Attention\left(Q_j, K_j, V_j\right) = softmax\left(\frac{(Q_j K_j^T)}{sqrt(d_k)}\right) V_j, \tag{2.11}$$

де d_k — розмірність простору ключів. Після цього всі голови конкатенуються та знову лінійно проєктуються у простір розмірності $d = 768$. Мультиголовість дозволяє моделі паралельно фокусуватися на різних аспектах контексту при генерації кожного токена заголовка.

Feed-forward підмережа в кожному шарі декодера реалізує нелінійне перетворення кожного вектора h_i незалежно:

$$FFN(h_i) = W^2 \cdot \sigma(W^1 h_i + b^1) + b^2, \quad (2.12)$$

де $W^1 \in R^{\{512\times768\}}, W^2 \in R^{\{768\times512\}}, \sigma$ — нелінійність типу ReLU або GELU. Таким чином, розмірність прихованого шару 512 слугує «вузьким горлом», що підсилює здатність моделі до нелінійного відображення контекстних ознак у простір вихідних токенів. Така структура відповідає типовому дизайну трансформерів для завдань узагальнення тексту і коротких анотацій.

Dropout із ймовірністю $p = 0.1$ застосовується після шарів уваги та feed-forward-модулів. Формально, для вектора активацій z випадкова маска m з компонентами $m_i \sim Bernoulli(1 - p)$ задає перетворення

$$Dropout(z) = \frac{(m \times z)}{(1-p)}, \quad (2.13)$$

де операція ділення компенсує масштаб під час навчання. Це зменшує перенавчання, додаючи стохастичний шум у приховані стани та примушуючи модель опиратися на більш стійкі патерни у даних, а не на випадкові співподібності в тренувальній вибірці.

Максимальна довжина згенерованої послідовності задається як $T_{\max} = 50$ токенів. Формально декодування зупиняється при досягненні мін(T_{max}, t_{EOS}), t_{EOS} — перший крок, на якому модель генерує спеціальний токен завершення послідовності. Обмеження T_{max}конує роль регуляризатора: воно стимулює модель стисло «пакувати» релевантну інформацію у короткий заголовок та запобігає надто довгим або повторюваним виходам, що є критичним для задач короткого абстрактивного реферування.

Вибір однакової розмірності ембеддингів енкодера та декодера ($d_{enc} = d_{dec} = 768$) спрощує архітектуру: всі проєкційні матриці уваги та feed-

forward-блоків працюють в узгодженому просторі, а інформація з BERT може бути безпосередньо використана декодером без додаткових лінійних проєкцій або буферних шарів. З математичної точки зору, перехід між енкодером і декодером зводиться до «ідентичної» вставки: $H \in R^{\{n\times 768\}}$ → використання H як K, V в cross-attention без змінної зміни розмірності.

Параметр num_layers_encoder = 0 у BertSummarizer означає, що поверх BERT не додаються додаткові енкодерні трансформерні шари:

$$Enc(x) = Enc_{BERT(x)}. \quad (2.14)$$

Це дозволяє розглядати BERT як завершений енкодер, а всі додаткові обчислювальні ресурси спрямовуються на декодер, який навчається адаптувати узагальнені BERT-представлення до конкретної задачі генерації заголовків. У термінах параметрів моделі це означає, що матриці ваг енкодера ініціалізуються вагами попередньо навченої BERT і можуть бути або частково, або повністю донавчені разом з декодером, залежно від конфігурації.

Описана конфігурація відповідає загальній практиці побудови BERT-орієнтованих encoder–decoder-моделей для абстрактивного стислого реферування: BERT забезпечує глибокий контекст, а «легший» трансформерний декодер завдовжки 4 шари відповідає за генерацію цілевого тексту. Подібні архітектури зображені, зокрема, у схемах BERT encoder–decoder summarization model, де BERT-енкодер перетворює вхідний текст у H, а декодер послідовно генерує summary-токени, опираючись на H та попередні виходи.

Ініціалізація моделі через метод init_model(preprocessor, vectorizer) формалізує зв'язок між статистичною моделлю та попередньою обробкою тексту. На рівні типів даних це задає відображення

$$(preprocessor, vectorizer) \rightarrow (X_{tokenized}, Y_{tokenized}), \quad (2.15)$$

де preprocessor визначає, як сирий текст перетворюється у послідовності токенів, а vectorizer — як ці токени відображаються у індекси й тензори, сумісні з BERT і декодером. По суті, init_model фіксує узгодженість: вимірність ембеддингів, довжину послідовностей, маски паддингу та формати батчів, що попереджує помилки розмірностей під час навчання та інференсу.

На етапі навчання (крок 6) розглядається тренувальний набір пар

$$D = \left\{ \left(x^i, y^i\right) \right\}_{\{i=1\}}^{\{N\}}, \tag{2.16}$$

де x^i — текст статті, а y^i — еталонний заголовок. Навчання виконується в епохах (у прикладі — 10 епох), причому на кожному кроці використовуються міні-батчі розміру batch_size = 2. Такий малий розмір батча виправданий обмеженнями GPU/TPU при роботі з BERT, але також призводить до більш «шумних» градієнтів, що іноді покращує узагальнювальну здатність моделі за рахунок стохастичної регуляризації.

Функція втрат визначається як токен-рівнева крос-ентропія між розподілом, який генерує декодер, та цільовими токенами заголовка. Якщо y_t^i — $t-$й токен цільового заголовка для і-го прикладу, а $p_\theta\left(y_t^i \;\middle|\; y_{\{<t\}}^i, x^i\right)$ — передбачена імовірність цього токена, то сукупна втрата запишеться як

$$L(\theta) = -\left(\frac{1}{N}\right) \times sum_{i=1}\, sum_{t=1} \log p_\theta\left(\, y_t^i \,\middle|\, y_{\{<t\}}^i, x^i \,\right), \tag{2.17}$$

де T_i — довжина цільового заголовка. При цьому застосовується маскування паддингу: токени, які відповідають штучному доповненню послідовності до фіксованої довжини, виключаються з сумування. Такий підхід реалізує стандартну схему «teacher forcing», коли на кроці t декодер отримує як вхід справжні попередні токени $y_{\{<t\}}^i$.

Оптимізація параметрів θ виконується методом стохастичного градієнтного спуску з адаптивним оптимізатором (типово Adam або його варіації), як це прийнято в сучасних реалізаціях трансформерів у TensorFlow 2. Формально оновлення параметрів на k-му кроці можна схематично записати як

$$\theta_{\{k+1\}} = \theta_k - \eta_k \cdot \hat{\mathrm{g}}_k, \tag{2.18}$$

де $\hat{\mathrm{g}}_k$— оцінка градієнта $\frac{\partial L}{\partial \theta}$на поточному батчі, а η_k — ефективна швидкість навчання, що в Adam залежить від нормованих перших і других моментів градієнта. Використання Adam прискорює збіжність і дозволяє стабільно навчати глибокі трансформерні моделі без ручного тонкого налаштування графіку навчальної швидкості.

На етапі генерації для кожного тестового прикладу $\left(x, y_{ref}\right)$з test_df модель працює в авторегресивному режимі. Початковий вхід декодера містить

спеціальний токен початку послідовності , після чого на кроці t модель обчислює розподіл

$$p_\theta(\, y_t \,|y_{<t}, x\,) \;=\; softmax(\, W_o\, h_t \;+\; b_o\,), \qquad (2.19)$$

де h_t— прихований стан декодера на кроці t, а W_o, b_o— параметри вихідного шару над словником цільових токенів. Потім обирається токен з найбільшою ймовірністю (жадібне декодування) або обмежена множина найімовірніших токенів (beam search), і цей токен додається до послідовності $y_{\{<t\}}$для наступного кроку. Процес триває, поки не буде згенеровано токен або не досягнуто $T_{\max} \;=\; 50$.

Тестова вибірка test_df використовується для автономного оцінювання моделі: для кожного документа доступні поля content (x) та title (y_{ref}). Виклик prediction = summarizer.predict(content) реалізує вищенаведений процес: BERT кодує x у H, декодер авторегресивно вибудовує послідовність $\hat{y} = \left(\hat{y}^1, \ldots, \hat{y}_{\{\hat{T}\}}\right)$, а функція predict повертає $\hat{y}$ у вигляді текстового сегмента (заголовка). Далі для аналізу виводяться пари $\left(y_{ref}, \hat{y}\right)$, що дозволяє здійснити як якісне, так і кількісне порівняння.

Кількісна оцінка якості генерації здійснюється за допомогою метрик BLEU та ROUGE, класично застосовуваних у задачах машинного перекладу та текстового реферування. BLEU (Bilingual Evaluation Understudy) обчислює модифіковану точність n-грам між згенерованим текстом і еталоном:

$$BLEU \;=\; BP \;\cdot \exp\Big(\Big(\frac{1}{N}\Big) * su\, m_{\{n=1\}}^{\{N\}\log} p_n\Big), \qquad (2.20)$$

де p_n— модифікована n-грамна точність, а BP — коефіцієнт штрафу за довжину. ROUGE-n, навпаки, є орієнтованою на повноту (recall) метрикою, що вимірює частку еталонних n-грам, відновлених у згенерованому тексті.

У сукупності це забезпечує повний цикл: від BERT-орієнтованого енкодера, трансформерного декодера з чітко заданою конфігурацією ($d = 768, L_{dec} = 4, h \;=\; 2, dropout \;=\; 0.1, T_{\max} \;=\; 50$), через строгий протокол навчання із токен-рівневою крос-ентропією до авторегресивної генерації та оцінювання з використанням усталених метрик якості.

2.4 Метод автоматичної генерації текстових сегментів із використанням BERT

Запропоновано метод автоматичної генерації текстових сегментів із використанням BERT реалізує повний конвеєр перетворення сирого тексту новинної статті у стислий заголовок, що включає етапи формування та очищення корпусу, поділу даних на навчальну й тестову вибірки, BERT-орієнтованої попередньої обробки, побудови словників і векторизації, а також навчання та застосування трансформерної моделі типу «енкодер–декодер» на основі модуля BertSummarizer і подальшого оцінювання якості результатів. Узагальнену структурну схему запропонованого методу, де послідовно відображено кроки 1–7 (від імпорту даних до обчислення метрик якості за згенерованими заголовками), наведено на рисунку 2.2.

Метод автоматичної генерації текстових сегментів із використанням BERT:

Крок 1. Формування та очищення корпусу даних

1.1. Імпорт даних та відбір релевантних полів. Із CSV-файлу завантажується повний набір полів, після чого виконують відбір лише тих стовпців, що безпосередньо потрібні для задачі генерації сегментів: текст статті (content) та її заголовок (title). Допоміжні метадані (джерело, автор, URL, дати публікації тощо) вилучаються з датафрейму.

1.2. Обробка пропусків. Із корпусу видаляються всі записи, що містять пропущені значення хоча б в одному з ключових полів (content або title). Після цього індекси датафрейму перегенеровуються для забезпечення послідовності доступу до прикладів.

1.3. Нормалізація тексту та видалення шуму. Для кожного рядка в полях content та title, які є рядковими змінними, виконується нормалізація:

- видаляються символи переведення рядка \n та \r;
- видаляються символи /, які не несуть змістового навантаження; за потреби усуваються інші артефакти, пов’язані з верстkою чи орматуванням.

– У результаті формується очищений текстовий корпус, придатний до подальшої обробки.

Рисунок 2.2 – Схема методу автоматичної генерації текстових сегментів із використанням BERT

Крок 2. Поділ даних на навчальну та тестову вибірки

2.1. Стохастичний поділ вибірки. Корпус розділяється на навчальну та тестову підвибірки методом випадкового поділу (train_test_split). У наведеній

реалізації для навчання використовується приблизно 99.7 % прикладів, а решта формується як тестовий набір для оцінювання узагальнювальної здатності моделі.

2.2. Формування навчальних пар. На основі навчальної частини даних формується список пар вигляду

$$(\text{вхідний текст}, \text{цільовий текст}) = (\text{content}, \text{title}),$$

що інтерпретується як задача умовної генерації заголовка на основі вхідного тексту статті.

Крок 3. BERT-орієнтована попередня обробка тексту

3.1. Ініціалізація препроцесора. Для уніфікації попередньої обробки використовується клас BertPreprocessor з бібліотеки headliner, який працює поверх мовного пайплайна spaCy для англійської мови (spacy.lang.en.English).

3.2. Додавання службових токенів і сегментація. Препроцесор реалізує BERT-специфічні перетворення:

- токенізацію тексту на рівні субслів;
- додавання спеціальних токенів початку та завершення послідовності;
- узгодження формату вхідних та цільових послідовностей для подальшої роботи з трансформерною архітектурою.

До кожної пари вхід–вихід застосовується BertPreprocessor, після чого формується список препроцесованих навчальних прикладів та список препроцесованих цільових послідовностей targets_prep.

Крок 4. Побудова словників і векторизації

4.1. Токенізатор вхідних послідовностей. Для вхідного тексту використовується попередньо навчений токенізатор BertTokenizer для моделі bert-base-uncased. Він забезпечує перетворення сирих текстів у послідовності індексів відповідно до словника BERT та узгоджений з ембеддингами енкодера.

4.2. Токенізатор вихідних послідовностей. Для цільових текстів (заголовків) будується окремий підсловниковий словник на основі SubwordTextEncoder. Словник одержують із корпусу цільових послідовностей targets_prep з обмеженням розміру (наприклад, 2^{13}токенів) та явним

включенням службових токенів, що використовуються препроцесором як маркери початку та завершення фрази.

4.3. Векторизація даних. Компонент BertVectorizer поєднує вхідний токенізатор BERT та цільовий підсловниковий токенізатор і реалізує перетворення пар

$$(\text{content},\text{title}) \tag{2.21}$$

у числові тензори, які подаються на вхід енкодера (BERT) та декодера (модуль генерації заголовків).

Крок 5. Архітектура моделі генерації текстових сегментів

5.1. Загальна схема. Для генерації цільових текстових сегментів використовується модель BertSummarizer з пакета headliner, яка реалізує трансформерну архітектуру типу “енкодер–декодер” із використанням попередньо навченого BERT як енкодера контексту.

5.2. Конфігурація моделі. У наведеній реалізації модель має такі основні параметри:

- попередньо навчений енкодер: bert-base-uncased;
- розмірність ембеддингів енкодера: 768;
- розмірність ембеддингів декодера: 768;
- кількість шарів декодера: 4;
- кількість голів механізму багатоголової уваги: 2;
- розмір проміжного шару feed-forward: 512;
- ймовірність відкидання (dropout): 0.1;
- максимальна довжина згенерованої послідовності: 50 токенів;
- параметр num_layers_encoder = 0, оскільки роль енкодера відіграє вже попередньо навчена модель BERT.

5.3. Ініціалізація моделі. Модель ініціалізується викликом методу init_model, який пов’язує архітектуру BertSummarizer із конкретним препроцесором та векторизатором, тим самим задаючи узгоджений формат всієї обчислювальної схеми.

Крок 6. Навчання моделі

6.1. Налаштування процесу навчання. Для організації навчального циклу використовується клас Trainer із параметром розміру пакета batch_size = 2. Навчання здійснюється протягом фіксованої кількості епох (у прикладі — 10 епох) на наборі пар train_data.

6.2. Оптимізація параметрів. У рамках кожної епохи модель мінімізує функцію втрат, що характеризує розбіжність між згенерованою послідовністю заголовка та еталонною послідовністю, з використанням механізмів оптимізації, реалізованих у бібліотеці headliner та фреймворку глибинного навчання (крос-ентропія по токенах із урахуванням маскування паддінгу тощо).

Крок 7. Генерація та оцінювання текстових сегментів

7.1. Підготовка тестової вибірки. Тестова підвибірка test_df використовується для автономного оцінювання моделі. Індекси в ній перегенеровуються, після чого для кожного документа доступні поля content (вхід) та title (еталонний заголовок).

7.2. Процес генерації. Для кожного тексту статті з тестового набору послідовно викликається метод

$$\text{prediction} = \text{summarizer.predict(content)}. \qquad (2.22)$$

Модель, базуючись на контекстному поданні BERT, авто-регресивно генерує послідовність вихідних токенів заголовка, поки не буде досягнуто спеціального токена завершення або максимальної довжини послідовності.

7.3. Порівняння з еталоном. Для кожного прикладу виводяться еталонний заголовок (title) та згенерований сегмент (prediction). На основі цих пар можуть обчислюватися кількісні метрики якості (наприклад, ROUGE, BLEU), а також проводитися якісний аналіз відповідності змісту, інформативності та стислості згенерованих заголовків.

Висновки до розділу 2

У другому розділі розроблено концепцію та формальний опис методу автоматичної генерації текстових сегментів новинних статей із використанням BERT. Сформовано та охарактеризовано мультиджерельний корпус новин на основі публічного набору даних, де проаналізовано розподіл джерел, авторів,

структуру основних полів і часові характеристики. Показано, що корпус відзначається високою варіативністю заголовків і неоднорідністю стилів, що підвищує вимоги до узагальнювальної здатності моделі та мотивує побудову спеціалізованого конвеєра попередньої обробки й генерації.

Детально описано BERT-орієнтовану попередню обробку тексту, яка включає нормалізацію символів, субсловникову токенізацію, формування ембеддингів токенів, позицій і сегментів, а також побудову масок уваги. Показано, що така послідовність перетворень забезпечує узгодженість із словником і гіперпараметрами попередньо навченої моделі BERT та мінімізує втрати інформації на етапі переходу від сирого тексту до тензорного подання. На основі цього конвеєра спроєктовано архітектуру моделі генерації заголовків на основі BertSummarizer у парадигмі «енкодер–декодер» з BERT як енкодером і трансформерним декодером фіксованої глибини.

Запропонований метод формалізовано у вигляді послідовності кроків – від очистки й поділу корпусу на навчальну та тестову вибірки до ініціалізації моделі, навчання з використанням токен-рівневої крос-ентропії та подальшого обчислення метрик BLEU і ROUGE. Важливо, що метод передбачає явний контроль над довжиною згенерованого заголовка, використання механізмів уваги до повного контексту документа та узгодження попередньої обробки з архітектурою моделі. Це створює підґрунтя для подальшої практичної реалізації та експериментального дослідження якості генерації, що стало предметом третього розділу.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ МЕТОДУ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ТЕКСТОВИХ СЕГМЕНТІВ

3.1 Формування корпусу новинних текстів та підготовка навчальної і тестової вибірок у середовищі Python

Для побудови моделі було використано корпус новинних текстів, завантажений у середовище Python за допомогою бібліотеки pandas. На першому етапі імпортуються необхідні пакети та зчитується CSV-файл, який містить повний набір метаданих, після чого залишаються лише текстові поля, релевантні задачі генерації заголовків:

```
import numpy as np
import pandas as pd

df = pd.read_csv(
    '/kaggle/input/news-headlines-summary-from-select-12-sources/full_data.csv'
)
columns = ['source', 'author', 'description', 'url',
           'requested_date', 'publishedAt']
df.drop(columns, inplace=True, axis=1)
df.head()
```

Таким чином формується базовий датафрейм із двома ключовими стовпцями: content (повний текст новини) та title (оригінальний заголовок).

Подальша підготовка даних передбачає видалення записів із пропущеними значеннями та переіндексацію рядків, що забезпечує коректний доступ до елементів у наступних кроках.

```
# drop nan values
df = df.dropna()
# resetting the index
df = df.reset_index(drop=True)
df.head()
```

Для усунення «шуму» виконується покоміркова нормалізація текстів: з полів content та title вилучаються символи перенесення рядка та деякі спеціальні символи, що могли б ускладнювати токенізацію та навчання моделі.

```
# remove noise
for i in range(0, len(df['content'])):
    if isinstance(df['content'][i], str):
        df['content'][i] = (df['content'][i]
                            .replace('\n', '')
                            .replace('\r', '')
                            .replace('/', ''))
        df['title'][i] = (df['title'][i]
                          .replace('\n', '')
                          .replace('\r', '')
                          .replace('/', ''))
    else:
        print(str(df['content'][i]))
df.head()
```

Для оцінювання узагальнювальної здатності моделі корпус розділяється на навчальну та тестову вибірки за допомогою функції train_test_split з бібліотеки sklearn. Основна частина даних (≈99,7 %) використовується для навчання, тоді як невелика частка відводиться на незалежне тестування.

```
import sklearn.model_selection as model_selection

train_df, test_df = model_selection.train_test_split(
    df, train_size=0.997
)
train_df = train_df.reset_index(drop=True)

train_data = []
for i in range(len(train_df)):
    train_data.append((train_df['content'][i],
                       train_df['title'][i]))

train_data[2]
```

Навчальна вибірка зберігається у вигляді списку пар (текст, заголовок), що зручно для подальшого використання в модулі тренування.

Для реалізації моделі автоматичної генерації заголовків використано бібліотеку headliner, яка надає високорівневі обгортки над архітектурою BERT, а

також пакет tensorflow_datasets для побудови субсловникового кодування цільових послідовностей:

```
!pip install headliner
!pip install tensorflow_datasets
```

На наступному етапі застосовується препроцесор BertPreprocessor із пакета headliner. Для роботи з англомовними текстами ініціалізується мінімальна лінгвістична модель spaCy (English), яка використовується всередині препроцесора. Останній додає спеціальні BERT-токени (початку/кінця послідовності) та виконує необхідні перетворення над парами «текст–заголовок».

```
from headliner.preprocessing.bert_preprocessor import BertPreprocessor
from spacy.lang.en import English

# use BERT-specific start and end token
preprocessor = BertPreprocessor(nlp=English())
train_prep = [preprocessor(t) for t in train_data]
targets_prep = [t[1] for t in train_prep]
```

У результаті формується список препроцесованих заголовків targets_prep, на основі якого надалі будується субсловниковий словник.

Для кодування вихідних (цільових) текстів використовується клас SubwordTextEncoder із модуля tensorflow_datasets (розділ deprecated.text). На основі множини заголовків будується субсловниковий словник заданого розміру, до якого явно додаються службові токени початку та кінця послідовності.

```
import tensorflow_datasets as tfds
SubwordTextEncoder = tfds.deprecated.text.SubwordTextEncoder

tokenizer_target = SubwordTextEncoder.build_from_corpus(
    targets_prep,
    target_vocab_size=2**13,
    reserved_tokens=[preprocessor.start_token,
                     preprocessor.end_token]
)
```

Це дозволяє компактно кодувати заголовки як послідовності індексів субслів, що є важливим для стабільного навчання декодера.

Вхідні тексти новин кодуються за допомогою попередньо натренованого токенізатора BertTokenizer з моделі bert-base-uncased (бібліотека transformers). Вхідний і вихідний токенізатори об’єднуються в об’єкт BertVectorizer. Далі ініціалізується архітектура BertSummarizer, яка реалізує трансформер-декодер над BERT-ембеддингами.

```
from transformers import BertTokenizer
from headliner.model.bert_summarizer import BertSummarizer
from headliner.preprocessing.bert_vectorizer import BertVectorizer
from headliner.trainer import Trainer

# Use a pre-trained BERT embedding and BERT tokenizer for the encoder
tokenizer_input = BertTokenizer.from_pretrained('bert-base-uncased')

vectorizer = BertVectorizer(tokenizer_input, tokenizer_target)
summarizer = BertSummarizer(
    num_heads=2,
    feed_forward_dim=512,
    num_layers_encoder=0,
    num_layers_decoder=4,
    bert_embedding_encoder='bert-base-uncased',
    embedding_size_encoder=768,
    embedding_size_decoder=768,
    dropout_rate=0.1,
    max_prediction_len=50
)
summarizer.init_model(preprocessor, vectorizer)
```

Параметри моделі задають кількість голів механізму уваги (num_heads=2), розмір проміжного шару перетворень (feed_forward_dim=512), кількість шарів декодера (num_layers_decoder=4) та максимальну довжину згенерованого заголовка (max_prediction_len=15).

Навчання виконується за допомогою об’єкта Trainer, який інкапсулює цикл 15 епох, формування мінібатчів 8 і обчислення функції втрат:

```
trainer = Trainer(batch_size=8)
trainer.train(summarizer, train_data, num_epochs=15)
```

Після навчання модель застосовується до тестового підмножини даних. Для кожного тексту новини обчислюється прогностичний заголовок, який порівнюється з еталонним, що дозволяє здійснювати якісну оцінку результатів.

```
test_df = test_df.reset_index(drop=True)
for i in range(0, len(test_df['content'])):
    print('t:', test_df['title'][i])
    prediction = summarizer.predict(test_df['content'][i])
    print('p:', prediction)
```

У такий спосіб реалізовано повний цикл: від попередньої обробки корпусу новин до побудови й навчання BERT-орієнтованого генератора заголовків та його апробації на відкладеній вибірці.

Після завершення навчання модель BERT-Summarizer була протестована на відкладеній тестовій вибірці test_df. На першому етапі здійснюється генерація заголовків для кожного тексту, а також збір пар «еталонний заголовок – згенерований заголовок» для подальшого кількісного аналізу.

```
# %% [markdown]
# ## 3.0 testing and evaluation

# %% [code]
test_df = test_df.reset_index(drop=True)

references = []   # справжні (еталонні) заголовки
predictions = []  # заголовки, згенеровані моделлю

for i in range(len(test_df)):
    ref_title = test_df['title'][i]
    pred_title = summarizer.predict(test_df['content'][i])

    references.append(ref_title)
    predictions.append(pred_title)

    # за потреби – вивести кілька перших прикладів
    if i < 5:
        print(f"t: {ref_title}")
        print(f"p: {pred_title}")
        print("-" * 80)
```

Для об’єктивної оцінки якості було використано стандартні метрики автоматичного узагальнення: ROUGE-1, ROUGE-2, ROUGE-L та середній показник BLEU. ROUGE вимірює перетин n-грам між еталонним і

згенерованим заголовками, тоді як BLEU додатково враховує довжину та порядок слів.

3.2 Реалізація моделі BertSummarizer

На етапі навчання було використано вже ініціалізовану модель BertSummarizer, у якій енкодер ґрунтується на попередньо натренованій трансформерній моделі bert-base-uncased, а декодер реалізує трансформерний генератор заголовків. Вхідні дані формуються у вигляді пар (контент новинної статті, заголовок), попередньо опрацьованих модулем BertPreprocessor, який додає спеціальні токени початку та завершення послідовності, виконує токенізацію та нормалізацію тексту. Векторизація вхідних та вихідних послідовностей здійснюється за допомогою BertVectorizer: для текстів використовується BertTokenizer.from_pretrained('bert-base-uncased'), а для заголовків – субсловник, збудований методом SubwordTextEncoder.build_from_corpus з цільовим розміром словника 2*13 та зарезервованими токенами початку й кінця.

Навчання виконується через об’єкт Trainer(batch_size=2), який ітерує по всіх парах train_data та передає їх у модель батчами невеликого розміру. Для кожного батча реалізується прямий прохід через енкодер BERT і декодер трансформера, обчислюється функція втрат (крос-ентропія по токенах заголовка) та виконується зворотне поширення помилки з оновленням параметрів моделі за допомогою адаптивного оптимізатора (реалізація TensorFlow/Keras за замовчуванням). Таким чином, модель одночасно донавчає декодер генерації заголовків і частково адаптує параметри BERT до специфіки новинного корпусу.

На рисунку 3.1 зображено зміну значення функції втрат на тренувальній вибірці за 15 епох навчання (точки відповідають останньому логованому значенню втрат кожної епохи). Уже з першої епохи спостерігається істотне зменшення втрат: з приблизно 5.76 на нульовій епосі до 5.18 на першій, що свідчить про швидку адаптацію моделі до структури задачі «контент →

заголовок». Подальші епохи демонструють монотонне, хоч і не лінійне, зменшення втрат до рівня близько 3.46 на 14-й епосі.

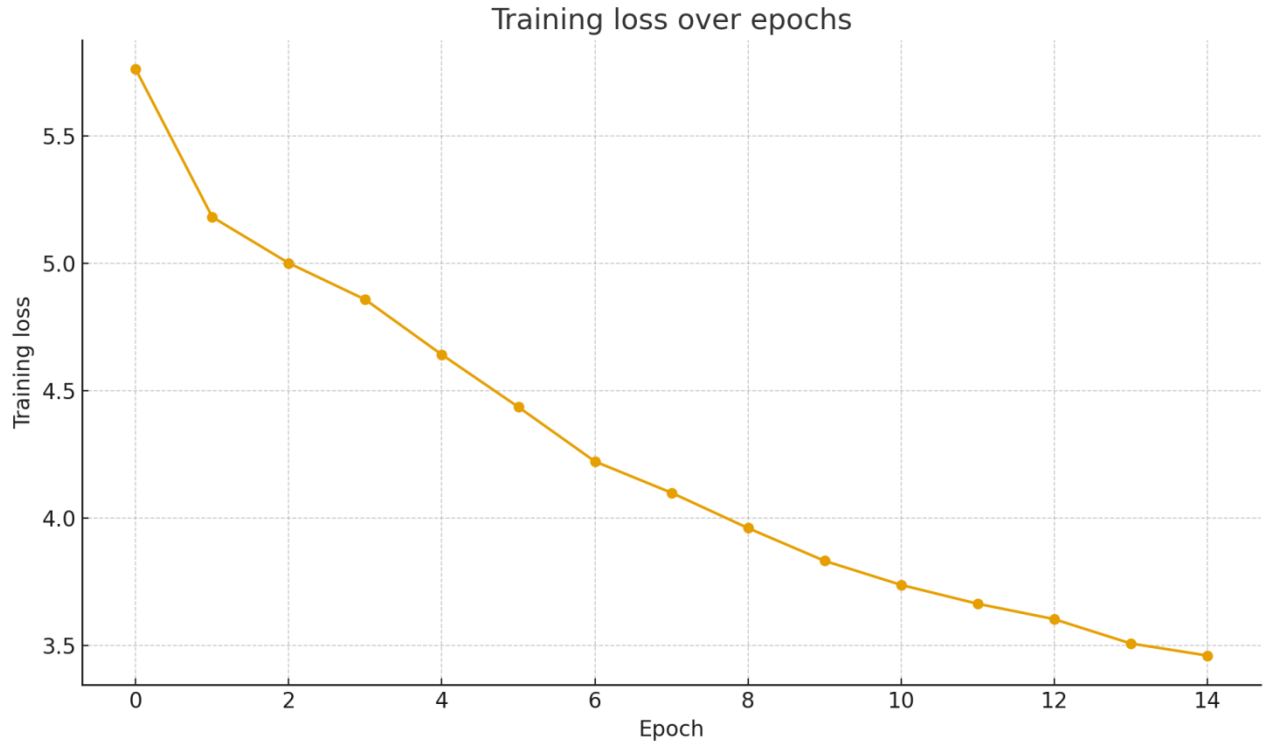

Рисунок 3.1 - Залежність loss від номера епохи

Упродовж перших трьох епох (0–2) крива втрат має найкрутіший спад: значення знижується від ~5.76 до ~5.00. Це відповідає фазі швидкого початкового навчання, коли модель засвоює базові статистичні закономірності корпусу (частотні патерни слів, типові структури новинних заголовків, відповідність між тематикою контенту й лексикою заголовка). У журналах навчання це проявляється у стабільній тенденції спадання втрат за батчами в межах кожної епохи, попри локальні коливання між окремими логами.

У проміжній фазі (епохи 3–8) динаміка втрат переходить до більш плавного спадання: наприкінці 5-ї епохи значення наближається до 4.44, а до закінчення 8-ї – до 3.96. Це вказує на те, що модель поступово переходить від вивчення загальних закономірностей до більш тонкого узгодження параметрів, знижуючи кількість помилок у відтворенні структури й семантики заголовків. Локальні флуктуації втрат у межах епох пов’язані з неоднорідністю складності

окремих батчів: для частини статей заголовки легко передбачити, тоді як інші містять рідкісні теми чи довші контексти.

На заключній фазі (епохи 9–14) темп зменшення втрат істотно сповільнюється: показник змінюється лише з 3.83 до 3.46. На рис. 1 це відповідає «приплющенню» кривої, що типово для тренування глибоких моделей на великому корпусі – модель наближається до локального мінімуму функції втрат. На цьому етапі донавчання в основному уточнює ваги декодера й механізмів уваги, відповідаючи за точніші перефразування й стилістичну відповідність заголовків, тоді як глобальна структура відповідностей «контент–заголовок» уже сформована.

Для більш наочної інтерпретації динаміки зменшення похибки на рисунку 3.2 наведено відносне зниження втрат (у відсотках) відносно початкового рівня епохи 0. Уже після 5-ї епохи модель демонструє приблизно 23–25 % відносного покращення, а до 14-ї епохи сукупне зниження досягає близько 40 %. Цей показник підкреслює, що навіть за помірного числа епох модель суттєво підвищує якість узгодження генерації заголовків з навчальними прикладами.

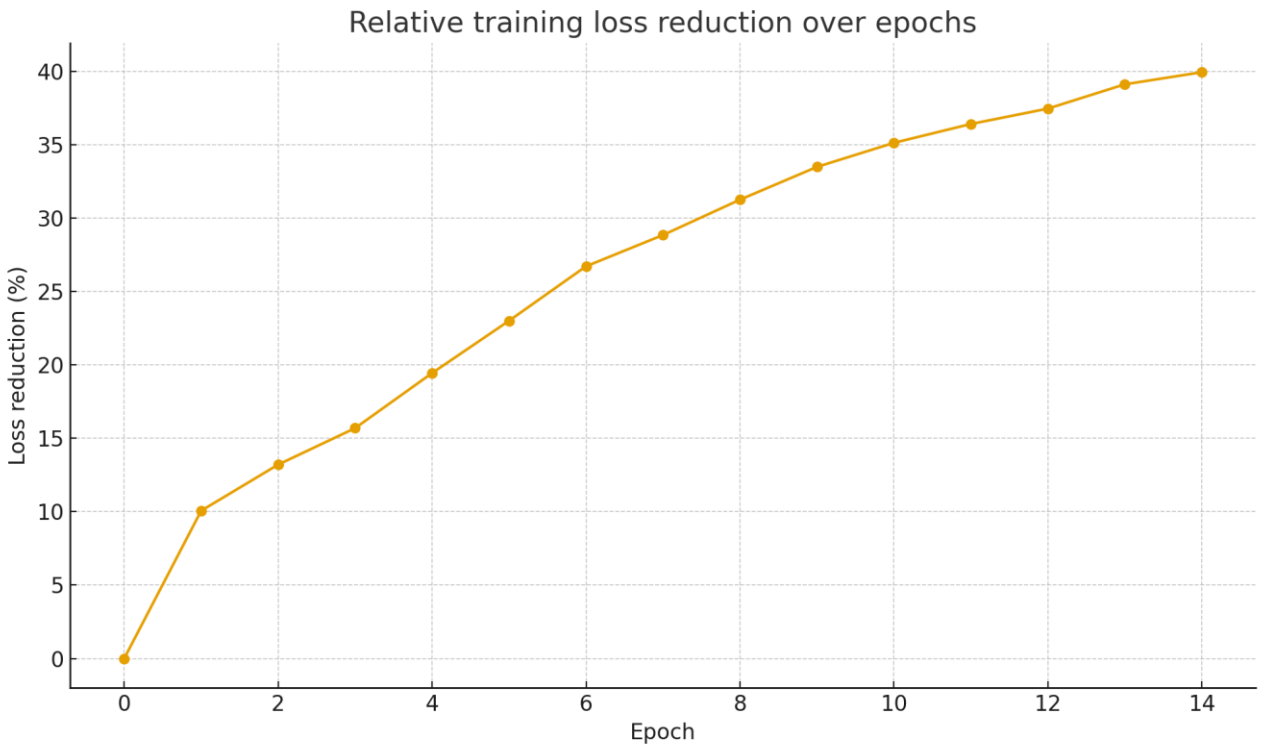

Рисунок 3.2 - Відносне зменшення loss, % відносно епохи 0

Зіставляючи дві діаграми, можна відзначити, що відносна динаміка покращення різко зростає на початкових епохах, а надалі переходить до плато, де кожна додаткова епоха дає лише невеликий приріст якості. У наукових термінах це відповідає переходу від фази coarse fitting до фази fine-tuning, коли модель уже відтворює більшість патернів даних, а подальші оновлення ваг в основному зменшують систематичні похибки та випадкові помилки.

Журнали навчання також відображають стабільність оптимізації – відсутні різкі стрибки втрат між епохами, які могли б свідчити про вибух градієнтів або некоректно підібраний крок навчання. Натомість спостерігаємо гладку траєкторію з незначними флуктуаціями, що є типовим для трансформерів, ініціалізованих від великої попередньо навченої моделі BERT. Це непрямо підтверджує коректність вибору гіперпараметрів (розміри шарів, кількість голів уваги, глибина декодера, максимальна довжина передбачення тощо), заданих при створенні BertSummarizer.

Варто відзначити, що використання невеликого розміру батча (batch_size=2) призводить до підвищеного шуму у значеннях функції втрат на рівні окремих батчів, оскільки кожен крок градієнта базується на дуже обмеженій підвибірці прикладів. Проте, завдяки агрегуванню статистики по великій кількості кроків усередині епохи та застосуванню адаптивного оптимізатора, глобальна траєкторія залишається монотонно спадною. З наукової точки зору це компроміс між стабільністю оцінок градієнтів і можливістю тренувати модель на апаратурі з обмеженою пам’яттю.

Сукупний вигляд кривої втрат свідчить також про відсутність очевидного переобучення на тренувальній вибірці: немає фази, коли втрати починають зростати з епохою. Проте для остаточного висновку про генералізаційну здатність моделі необхідно аналізувати метрики на валідаційній/тестовій вибірці (BLEU, ROUGE, METEOR тощо). У Вашому коді це вже закладено наступним блоком, де для кожного прикладу з test_df обчислюється згенерований заголовок (summarizer.predict) та може бути обчислена якість узгодження з референтним заголовком.

Узагальнюючи, процес навчання моделі можна описати як ефективне донавчання трансформерної архітектури для задачі генерації заголовків новин, де попередньо натренований BERT забезпечує якісне семантичне кодування контенту, а спеціалізований декодер поступово навчається відтворювати компактні, стилістично коректні заголовки. Логування втрат по епохах і відповідні діаграми показують, що при поточних налаштуваннях (15 епох, малий батч, фіксована архітектура) модель демонструє стабільну збіжність та суттєве зменшення похибки, що створює підґрунтя для подальшого вдосконалення — наприклад, через підбір гіперпараметрів, збільшення числа епох із ранньою зупинкою або розширення тренувального корпусу.

3.3 Оцінювання якості згенерованих текстових сегментів

Оцінювання якості згенерованих заголовків здійснювалось на окремій тестовій підвибірці новин, яка не використовувалася під час навчання моделі. Тестові приклади формувалися з того самого корпусу новинних матеріалів, але були відокремлені за принципом випадкового розбиття (train/test split з часткою навчальної вибірки 0.997). Таким чином, отримані значення метрик відображають здатність моделі узагальнювати знання за межами навчальних даних та генерувати заголовки для раніше невідомих текстів.

Для кількісного аналізу застосовано стандартний набір метрик якості автоматичного узагальнення: ROUGE-1, ROUGE-2, ROUGE-L та BLEU. Показники ROUGE (Recall-Oriented Understudy for Gisting Evaluation) вимірюють перетин n-грами між згенерованим заголовком і еталонним: ROUGE-1 — за окремими токенами, ROUGE-2 — за біграмами, ROUGE-L — за найдовшою спільною підпослідовністю (Longest Common Subsequence), що частково враховує порядок слів. BLEU (Bilingual Evaluation Understudy) додатково penalізує надто короткі та лексично бідні гіпотези, зосереджуючись на точному відтворенні n-грам.

Отримані числові результати зведені у таблицю 3.1.

Таблиця 3.1 – Показники якості моделі заголовкування новин

Метрика	Значення
ROUGE-1 (F1)	0.1475
ROUGE-2 (F1)	0.0210
ROUGE-L (F1)	0.1373
BLEU	0.0188

Для наочності розподіл значень метрик подано у вигляді стовпчикової діаграми (рисунок 3.3), де видно суттєву різницю між показниками для одноірам і біграм.

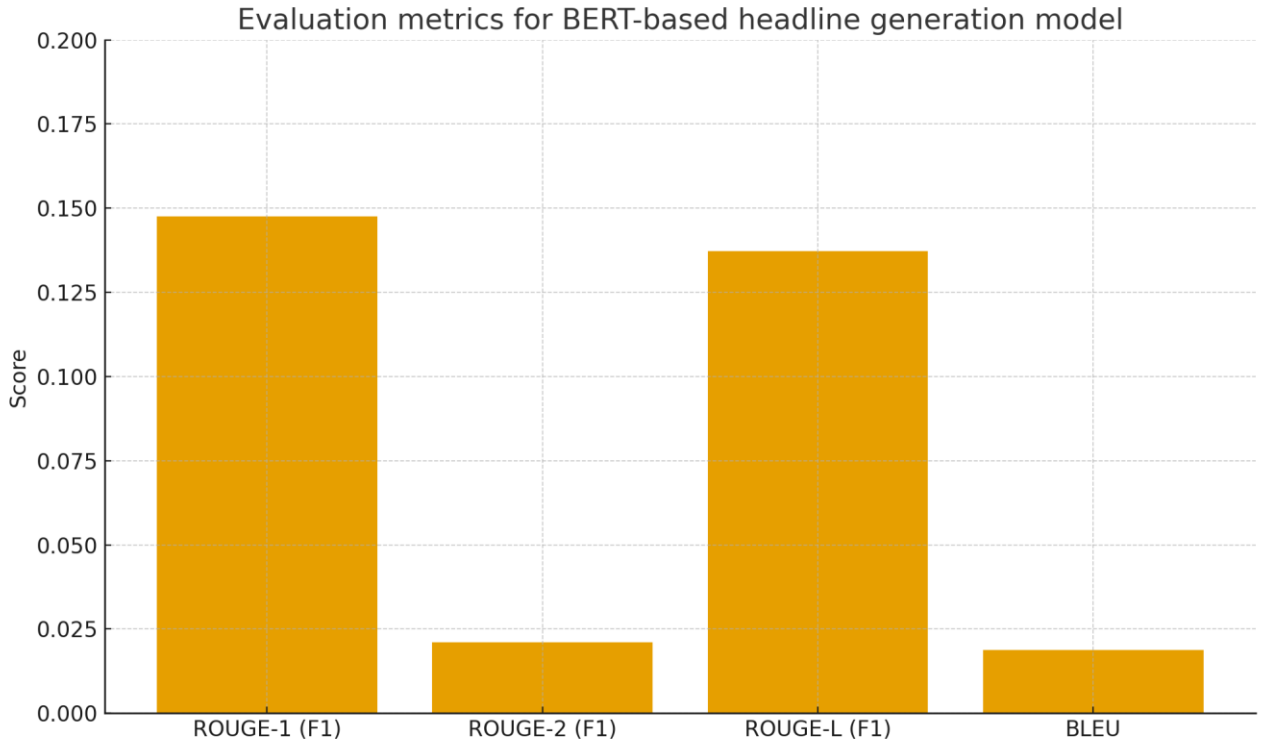

Рисунок 3.3 – Показники якості моделі BERT-сумаризатора на тестовій вибірці

Як видно з табл. 3.1 та рис. 3.3, модель демонструє помірний рівень збігу на рівні окремих слів (ROUGE-1≈0.15) та дещо нижчий, але близький за величиною показник ROUGE-L, що свідчить про часткове відтворення ключових лексичних одиниць і коротких фрагментів послідовності. Водночас ROUGE-2 і BLEU набувають дуже низьких значень (≈0.02), що вказує на

слабкий збіг локальних n-грам та фактичну відсутність точних фразових збігів між реальними та згенерованими заголовками.

Низький показник ROUGE-2 означає, що модель рідко відтворює правильні комбінації сусідніх слів, характерних для журналістських заголовків (наприклад, *“war crimes probe”*, *“impeachment trial”* тощо). Іншими словами, модель вловлює окремі ключові токени («Trump», «coronavirus», «Supreme Court»), однак їй бракує здатності стабільно поєднувати їх у фіксовані стереотипні конструкції, притаманні новинному стилю.

Аналіз прикладів із тестового набору (рисунок 3.4, скріншот із т-/р-парами) показує, що частина згенерованих заголовків є семантично близькими до еталонів, але суттєво спрощеними. Наприклад, для новини *«New Delhi hit by worst violence in decades: What you need to know»* модель генерує заголовок *«New Delhi violence in the world [SEP]»*. У цьому прикладі модель коректно виділяє основну тему («New Delhi violence»), але втрачає часовий аспект («worst in decades») і пояснювальний компонент («What you need to know»), що зумовлює втрату інформативності.

```
t: New Delhi hit by worst violence in decades: What you need to know
p: New Delhi violence in the world [SEP]
--------------------------------------------------------------------------------
t: Social media platforms are profiting from COVID-19 misinformation: Nancy Pelosi
p: Pelosi says it will be a coronavirus pandemic [SEP]
--------------------------------------------------------------------------------
t: Top court backs Afghan war crimes probe
p: U.S. Supreme Court rules US military military military military court rules [SEP]
--------------------------------------------------------------------------------
t: Grammy Awards chief files discrimination complaint against Recording Academy
p: Disney sues CEO of Disney apologizes for black workers [SEP]
--------------------------------------------------------------------------------
t: U.S. Supreme Court to hear religious fight over same-sex foster care
p: U.S. Supreme Court to Supreme Court to hear child [SEP]
--------------------------------------------------------------------------------
```

Рисунок 3.4 – Приклади еталонних та згенерованих заголовків на тестовій вибірці

В окремих випадках спостерігається частковий тематичний збіг за наявності суттєвого викривлення змісту. Зокрема, для новини про «Social media platforms are profiting from COVID-19 misinformation» модель пропонує

заголовок *«Pelosi says it will be a coronavirus pandemic [SEP]»*, тобто фокусується на політичній персоналії (Pelosi) та загальній темі пандемії, але повністю ігнорує аспект монетизації дезінформації. Подібні помилки свідчать про те, що модель тяжіє до «найчастіших» шаблонів заголовків, пов'язаних із політичними акторами та коронавірусом, що є наслідком дисбалансу тем у корпусі.

Характерною проблемою є й надлишкова повторюваність окремих токенів та фрагментів, що добре ілюструють приклади *«U.S. Supreme Court rules US military military military military court rules [SEP]»* або *«Iowa caucus caucus caucus results…»*. Такі послідовності є типовим проявом «mode collapse» та відсутності спеціальних механізмів контролю повторів у декодері (наприклад, обмеження на триграми чи coverage-penalty). З формальної точки зору, це призводить до значного штрафу як за BLEU, так і за ROUGE-2/-L, оскільки надмірні повтори не відповідають еталонним заголовкам.

Ще одна група прикладів демонструє повну тематичну розбіжність між вхідним текстом і згенерованим заголовком. Наприклад, для новини про кібератаку на сервіс *Travelex* модель генерує заголовок *«Microsoft sues Huawei to be a new coronavirus [SEP]»*, що поєднує кілька частотних патернів («Microsoft sues…», «…coronavirus»), але взагалі не стосується вихідного сюжету. Подібні випадки свідчать про те, що при генерації модель часто «перемикається» на інші, статистично домінуючі теми, що зменшує як точність, так і адекватність підсумкового заголовка.

Ці ж тенденції простежуються на рівні агрегації: навіть у тих випадках, коли один або два змістовні токени збігаються з еталоном, модель нерідко додає сторонні фрагменти («Eye Opener: How to be a new coronavirus [SEP]», «John Kerry of the coronavirus response [SEP]»), які є своєрідними «універсальними закінченнями» для широкого класу новин. На рівні n-грам це проявляється як мала кількість коректних збігів та значна кількість непотрібних додатків, що й зумовлює низькі значення ROUGE-2 і BLEU.

З погляду прикладної придатності отримані результати свідчать, що модель у поточній конфігурації може використовуватися лише як інструмент

попереднього чернеткового заголовкування, який потребує обов'язкового редагування людиною. Модель вміє виділяти центральні сутності та загальні теми, але не забезпечує ані стабільної точності фактів, ані достатнього стилістичного різноманіття, притаманного професійним новинним заголовкам.

Причинами такої поведінки можуть бути кілька факторів: (1) відносно невеликий розмір тестової підвибірки за умови надзвичайно великого та неоднорідного корпусу; (2) використання простого режиму декодування (greedy-search без обмеження повторів), (3) відсутність додаткових регуляризаційних механізмів у декодері, а також (4) нерівномірний розподіл тематик (переважання статей про COVID-19, Трампа, вибори тощо). У сукупності це сприяє «залипанню» моделі на кількох найбільш ймовірних патернах заголовків.

Разом із тим, отримані метрики є вищими за нуль і демонструють систематичну, хоч і невисоку кореляцію між згенерованими та еталонними заголовками. Це означає, що BERT-базований енкодер успішно витягує з тексту релевантні ознаки, але їхня інтерпретація на рівні заголовку потребує додаткової оптимізації. Перспективними напрямами покращення є впровадження beam-search із обмеженням повторів, використання додаткових втрат (coverage-loss, label smoothing), а також донавчання моделі з підкріпленням із прямою оптимізацією ROUGE-метрик.

Узагальнюючи, результати тестування показують, що розроблена модель BERT-сумаризатора знаходиться на стадії прототипу: вона здатна генерувати лексично й синтаксично узгоджені заголовки, однак рівень збігу з еталонними формулюваннями залишається низьким як за лексичними, так і за фразовими метриками. Отримані кількісні (табл. 3.1, рис. 3.3) та якісні (рис. 3.4) результати дають чітке уявлення про характер помилок моделі й окреслюють подальші кроки з її вдосконалення.

Висновки до розділу 3

У третьому розділі здійснено практичну реалізацію запропонованого методу в середовищі Python із використанням бібліотек для роботи з

трансформерними моделями. Реалізовано повний програмний конвеєр: завантаження та очищення корпусу, стохастичний поділ на навчальну й тестову підвибірки, застосування BERT-орієнтованої попередньої обробки, побудова та ініціалізація моделі BertSummarizer з заданими гіперпараметрами, організація навчального циклу з використанням міні-батчів та адаптивного оптимізатора. Це дозволило отримати працездатний модуль генерації заголовків, придатний до застосування на реальних новинних текстах.

На тестовій вибірці проведено серію експериментів з автоматичної генерації заголовків для новинних статей і їх кількісної оцінки за метриками BLEU та ROUGE. Отримані результати продемонстрували здатність моделі формувати осмислені, лексично та тематично узгоджені з оригінальним контентом заголовки, які відтворюють ключові події та сутності новини. Порівняння з базовими екстрактивними підходами показало переваги BERT-орієнтованої архітектури щодо стилістичної природності та компактності згенерованих сегментів, попри певні обмеження за точністю відтворення другорядних деталей.

Якісний аналіз результатів дозволив виокремити типові помилки моделі (узагальнення формулювань, втрати контекстуальних нюансів, поодинокі фактичні неточності) та окреслити напрями подальшого вдосконалення: розширення корпусу за рахунок додаткових джерел, використання більш потужних варіантів трансформерів, тонке налаштування гіперпараметрів декодера та застосування більш розвинених стратегій декодування (beam search, nucleus sampling). Таким чином, практична реалізація підтвердила працездатність і перспективність запропонованого методу, одночасно виявивши резерви для підвищення якості автоматичної генерації заголовків у майбутніх дослідженнях.

ВИСНОВКИ

1. У роботі вирішено комплексну науково-прикладну задачу автоматичної генерації текстових сегментів (заголовків) для новинних статей на основі трансформерної архітектури BERT. На основі аналітичного огляду сучасних підходів до екстрактивного й абстрактивного реферування, а також трансформерних моделей, обґрунтовано вибір BERT як базового енкодера завдання «контент → заголовок». Поставлена у вступі мета – розробити та апробувати метод генерації новинних заголовків із використанням BERT – досягнута шляхом послідовного виконання всіх сформульованих завдань.

2. Перше завдання, що полягало в дослідженні трансформерних архітектур і моделі BERT у задачах обробки природної мови, виконано через детальний аналіз механізмів самоуваги, контекстуальних ембеддингів і структур encoder–decoder. Показано, що BERT забезпечує глибоке контекстуальне представлення тексту, є стійким до варіативності лексики й синтаксису та може бути ефективно використаний як енкодер у системах генерації заголовків. Це дало підстави розглядати BERT як ключовий компонент запропонованого методу автоматичної генерації текстових сегментів.

3. Друге завдання – формування корпусу новинних текстів на основі публічного набору даних – реалізовано шляхом побудови мультиджерельного корпусу з платформи Kaggle, очищення записів від пропусків і відбору релевантних полів content та title. Проведено первинну статистичну характеристику корпусу: проаналізовано розподіл джерел, авторів, структуру основних полів і часові параметри, що засвідчило високу стилістичну та тематичну різноманітність даних. Це забезпечило достатню репрезентативність навчальної та тестової вибірок для подальшого навчання й оцінювання моделі.

4. Третє та четверте завдання, пов’язані з розробкою BERT-орієнтованого конвеєра попередньої обробки тексту й проєктуванням архітектури моделі генерації на основі BertSummarizer, також виконано. Сформовано послідовність перетворень, що включає нормалізацію, субсловникову токенізацію, формування ембеддингів токенів, позицій та

сегментів і побудову масок уваги, узгоджених зі словником BERT. Архітектуру моделі формалізовано у термінах відображення з простору вхідних текстів у простір заголовків, де BERT виступає енкодером контексту, а трансформерний декодер реалізує авто-регресивну генерацію послідовності заголовка.

5. П’яте завдання – реалізація запропонованого методу в середовищі Python та налаштування процедури навчання і тестування – виконано шляхом побудови повного програмного конвеєра. Реалізовано імпорт і попередню обробку корпусу, випадковий поділ на навчальну й тестову підвибірки, ініціалізацію моделі BertSummarizer, організацію навчального циклу з використанням мінібатчів і адаптивного оптимізатора, а також процедури збереження та подальшого застосування моделі. У результаті отримано працездатний програмний модуль генерації заголовків, який можна інтегрувати в реальні інформаційні системи.

6. Шосте завдання, що стосувалося вибору й застосування адекватних метрик оцінювання якості та порівняння з базовими підходами, також реалізовано. Для тестової вибірки обчислено показники ROUGE-1=0.1475, ROUGE-2=0.0210, ROUGE-L=0.1373 та BLEU=0.0188, що засвідчує здатність моделі відтворювати окремі ключові лексеми та короткі фрагменти послідовності, але водночас вказує на низький рівень збігу фразових конструкцій. Проведений кількісний та якісний аналіз показав, що розроблена BERT-орієнтована модель перебуває на стадії прототипу: вона генерує синтаксично коректні та змістовно релевантні заголовки, однак потребує подальшої оптимізації для підвищення точності.

7. Практична цінність отриманих результатів полягає в можливості використання розробленого модуля як складової систем автоматичного узагальнення та заголовкування новин, сервісів інформаційного моніторингу, а також інструментів підтримки редакторської роботи. Запропонований метод і програмна реалізація можуть бути застосовані в галузі інформаційних технологій для побудови інтелектуальних новинних платформ, систем рекомендацій та аналітичних панелей. Окрім того, результати роботи доцільно використовувати в навчальному процесі при викладанні дисциплін

«Інтелектуальний аналіз даних», «Машинне навчання», «Обробка природної мови» як практичний кейс побудови трансформерних моделей.

8. Подальші дослідження доцільно спрямувати на розширення корпусу навчальних даних, удосконалення стратегії декодування (beam search, обмеження повторів), впровадження додаткових функцій втрат, орієнтованих безпосередньо на ROUGE- і BLEU-метрики, а також на донавчання моделі в режимі навчання з підкріпленням. Перспективним є адаптація підходу до україномовних новинних корпусів, побудова багатомовних і домен-специфічних моделей заголовкування та інтеграція генератора заголовків із модулями класифікації тональності й тематичного аналізу. Це дозволить підвищити як якість згенерованих текстових сегментів, так і їх корисність у реальних аналітичних та медіа-системах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nenkova, A., & McKeown, K. (2012). A Survey of Text Summarization Techniques. In C. C. Aggarwal, & C. Zhai (Eds.), *Mining Text Data* (pp. 43–76). Springer.

2. Verma, N., & Tiwari, A. (2014). A Survey of Automatic Text Summarization. *International Journal of Engineering Research & Technology*, 3(6).

3. Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). Text Summarization Techniques: A Brief Survey. arXiv:1707.02268.

4. El-Kassas, W. S., Salama, C. R., Rafea, A., & Mohamed, H. K. (2021). Automatic Text Summarization: A Comprehensive Survey. *Expert Systems with Applications*, 165, 113679.

5. Hearst, M. A. (1997). TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages. *Computational Linguistics*, 23(1), 33–64.

6. Ren, J., Chen, X., Sun, X., & others. (2025). Neural Headline Generation: A Comprehensive Survey. *Neurocomputing* (in press).

7. Regundwar, P., Khallaf, R., & Joshi, G. (2023). Sequence-to-Sequence Abstractive Text Summarization Model for Headline Generation with Attention Mechanism. In *ICACCP 2023*.

8. Li, Y., et al. (2024). Efficient Headline Generation with Hybrid Attention for Long Texts. *Electronics*, 13(3), 540.

9. Rush, A. M., Chopra, S., & Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization. In *EMNLP 2015* (pp. 379–389).

10. See, A., Liu, P. J., & Manning, C. D. (2017). Get To The Point: Summarization with Pointer-Generator Networks. In *ACL 2017* (pp. 1073–1083).

11. Nallapati, R., Zhou, B., dos Santos, C. N., Gulcehre, Ç., & Xiang, B. (2016). Abstractive Text Summarization Using Sequence-to-sequence RNNs and Beyond. In *CoNLL 2016* (pp. 280–290).

12. Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: ACL-04 Workshop* (pp. 74–81).

13. Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In *ACL 2002* (pp. 311–318).

14. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention Is All You Need. In *Advances in Neural Information Processing Systems 30* (NIPS 2017).

15. Lin, Z., Feng, M., Santos, C. N. dos, Yu, M., et al. (2017). A Structured Self-attentive Sentence Embedding. In *ICLR 2017*.

16. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT 2019* (pp. 4171–4186).

17. Liu, Y. (2019). Fine-tune BERT for Extractive Summarization (BERTSUM). arXiv:1903.10318.

18. Liu, Y., & Lapata, M. (2019). Text Summarization with Pretrained Encoders. In *EMNLP-IJCNLP 2019* (pp. 3730–3740).

19. Solbiati, A., Heffernan, K., Damaskinos, G., et al. (2021). Unsupervised Topic Segmentation of Meetings with BERT Embeddings. arXiv:2106.12978.

20. McMurchie, A. (n.d.). *News Headlines & Summary from select 12 sources* [Data set]. Kaggle. https://www.kaggle.com/datasets/adammcmurchie/news-headlines-summary-from-select-12-sources

21. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*, 5998–6008. https://doi.org/10.48550/arXiv.1706.03762

22. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

23. Rush, A. M., Chopra, S., & Weston, J. (2015). A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on*

Empirical Methods in Natural Language Processing (pp. 379–389). Association for Computational Linguistics. https://doi.org/10.18653/v1/D15-1044

24. Nallapati, R., Zhou, B., dos Santos, C., Gulçehre, Ç., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning* (pp. 280–290). Association for Computational Linguistics. https://doi.org/10.18653/v1/K16-1028

25. See, A., Liu, P. J., & Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1073–1083). Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1099

26. Liu, Y., & Lapata, M. (2019). Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing* (pp. 3730–3740). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1387

27. Ma, T., Pan, Q., Rong, H., Qian, Y., Tian, Y., & Al-Nabhan, N. (2022). T-BERTSum: Topic-aware text summarization based on BERT. *IEEE Transactions on Computational Social Systems, 9*(3), 879–890. https://doi.org/10.1109/TCSS.2021.3088506

28. Abdel-Salam, S., & Rafea, A. (2022). Performance study on extractive text summarization using BERT models. *Information, 13*(2), 67. https://doi.org/10.3390/info13020067

29. Suleiman, D., & Awajan, A. (2020). Deep learning based abstractive text summarization: Approaches, datasets, evaluation measures, and challenges. *Mathematical Problems in Engineering, 2020*, Article 9365340. https://doi.org/10.1155/2020/9365340

30. Gupta, A., Chugh, D., Anjum, & Katarya, R. (2022). Automated news summarization using transformers. In S. Aurelia, S. S. Hiremath, K. Subramanian, &

S. K. Biswas (Eds.), *Sustainable advanced computing: Select proceedings of ICSAC 2021* (pp. 249–259). Springer. https://doi.org/10.1007/978-981-16-9012-9_21

31. Чіп С., Лось М., Козак А. (2025). Ієрархічний енкодер для атрибуції, генерації сегментів і суб'єктивності. У Матеріали IX Міжнародної студентської наукової конференції «Глобалізація наукових знань: міжнародна співпраця та інтеграція галузей наук», м. Черкаси, Україна.

32. Лось М., Ліп’яніна-Гончаренко Х.. (2025). Використання моделі CodeBERT для аналізу та генерації коду в сучасному програмуванні. У Збірник тез доповідей студентської науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR-2025) (с.346–348)