

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп'ютерних систем**

«На правах рукопису»  
УДК 004.91

«До захисту допущено»  
Завідувач кафедри  
\_\_\_\_\_ Євгенія СУЛЕМА  
«\_\_» \_\_\_\_\_ 2025 р.

**Магістерська дисертація**

**на здобуття ступеня магістра**

**за освітньо-науковою програмою «Інженерія програмного  
забезпечення мультимедійних та інформаційно-пошукових систем»**

**спеціальності 121 Інженерія програмного забезпечення**

**на тему: «Метод та програмне забезпечення для узагальнення  
великорозмірних науково-популярних текстів»**

Виконав:

студент II курсу, групи КП-31мн  
Кінаш Дарій Олегович \_\_\_\_\_

Керівник:

доцент кафедри ПЗКС, к.т.н., доцент,  
Заболотня Тетяна Миколаївна \_\_\_\_\_

Консультант з нормоконтролю:

доцент кафедри ПЗКС, к.т.н., доцент,  
Онай Микола Володимирович \_\_\_\_\_

Рецензент:

доцент кафедри СПіСКС, к.т.н., доцент,  
Тарасенко-Клятченко Оксана Володимирівна \_\_\_\_\_

Засвідчую, що у цій магістерській  
дисертації немає запозичень з праць  
інших авторів без відповідних  
посилань.

Студент \_\_\_\_\_

Київ – 2025 року

**Національний технічний університет України**  
**«Київський політехнічний інститут імені Ігоря Сікорського»**  
**Факультет прикладної математики**  
**Кафедра програмного забезпечення комп'ютерних систем**

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-наукова програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_ Євгенія СУЛЕМА

«\_\_» \_\_\_\_\_ 2023 р.

**ЗАВДАННЯ**  
**на магістерську дисертацію студенту**

Кінашу Дарію Олеговичу

1. Тема дисертації «Метод та програмне забезпечення для узагальнення великорозмірних науково-популярних текстів», науковий керівник Заболотня Тетяна Миколаївна, к.т.н., доцент, затверджені наказом по університету №1219-С від 21.03.2025 р.
2. Термін подання студентом дисертації 16 травня 2025 р.
3. Об'єкт дослідження: процес автоматизованого аналізу текстових даних.
4. Предмет дослідження: методи, способи, програмні засоби узагальнення природномовного тексту.
5. Перелік завдань, які потрібно розробити:
  - дослідити та проаналізувати наявні методи узагальнення великорозмірного тексту, зосередившись на швидкості їхньої роботи та якісних характеристиках їхнього результату узагальнення;
  - обрати еталонний метод для порівняння ефективності, що забезпечує зв'язність, послідовність та охоплення ключових тем вхідного тексту;
  - дослідити характерні риси науково-популярних текстів;
  - розробити метод для узагальнення великорозмірних науково-популярних текстів з покращеною швидкістю роботи порівняно з обраним еталонним методом;
  - обрати програмні інструменти для реалізації запропонованого методу;
  - створити програмне забезпечення, яке реалізує метод;
  - провести експерименти для оцінювання ефективності розробленого методу.

## 6. Перелік графічного (ілюстративного) матеріалу:

- діаграма етапів виконання запропонованого методу;
- діаграма етапів виконання еталонного методу для порівняння ефективності;
- діаграма алгоритму абстрактного узагальнення з застосуванням фрагментації;
- діаграма модулів розробленого програмного забезпечення;
- діаграма сценаріїв використання розробленого програмного забезпечення;
- діаграми порівняння швидкості роботи запропонованого та еталонного методів.

## 7. Орієнтовний перелік публікацій:

- тези доповіді “Комбінований метод узагальнення великорозмірних текстів” на конференції «Прикладна математика та комп’ютинг. ПМК-2024»

## 8. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент кафедри ПЗКС		

## 9. Дата видачі завдання «10» жовтня 2023 р.

### Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів проєкту	Примітка
1.	Ознайомлення з предметною галуззю	20.10.2023	
2.	Проведення досліджень для знаходження рішень	31.08.2024	
3.	Підготовка матеріалів доповіді на ПМК-2024 за проміжними результатами дослідження	10.11.2024	
4.	Розроблення методу для узагальнення великорозмірних науково-популярних текстів	31.12.2024	
5.	Створення програмного забезпечення для реалізації методу	31.01.2025	
6.	Проведення експериментів для оцінювання методу	28.02.2025	
7.	Написання статті за результатами проведених експериментів	31.03.2025	
8.	Оформлення текстової і графічної частини магістерської дисертації	04.05.2025	

Студент

Дарій КІНАШ

Науковий керівник

Тетяна ЗАБОЛОТНЯ

## РЕФЕРАТ

**Актуальність теми:** Станом на сьогодні існує низка рішень у галузі автоматизованого узагальнення текстів, зокрема як екстрактного, так і абстрактного типу. Ці методи дозволяють скорочувати природномовні тексти до стислих викладів, зберігаючи при цьому їхню основну суть. Проте більшість сучасних підходів до узагальнення тексту демонструють високу ефективність переважно на коротких текстах, таких як новинні статті. З часом були розроблені методи обробки великорозмірних текстів, однак вони потребують значних обчислювальних ресурсів та багато часу для створення якісного результату узагальнення. Це зумовлено як обсягом текстового матеріалу, так і необхідністю аналізу змісту, стилістичних особливостей і логіки викладу тексту. У випадку з великорозмірними науково-популярними текстами, які набули великої популярності в останні десятиліття та зазвичай мають широкий тематичний діапазон, ці підходи не можуть забезпечити належну швидкість та якість узагальнення. Тому актуальним завданням є розробка методу узагальнення, здатного ефективно працювати з великорозмірними науково-популярними текстами, при цьому зберігаючи в результаті узагальнення зв'язність, послідовність та охоплення ключових тем вхідного тексту.

**Об'єктом дослідження** є процес автоматизованого аналізу природномовних даних.

**Предметом дослідження** є методи, способи, програмні засоби узагальнення природномовного тексту.

**Метою дослідження** є підвищення швидкості процесу узагальнення великорозмірних науково-популярних текстів при збереженні таких якісних характеристик результату узагальнення, як зв'язність, послідовність та охоплення ключових тем.

**Наукова новизна.** Вперше запропоновано метод для узагальнення великорозмірних науково-популярних текстів, що створює зв'язний та

послідовний результат, який охоплює ключові теми вхідного тексту, та має як мінімум в 1.8 разів більшу швидкість роботи, ніж багатокроковий метод абстрактного узагальнення з застосуванням фрагментації визначеної контекстом.

**Практична значущість.** Розроблений метод може слугувати основою для подальших досліджень у галузі обробки природномовних даних і для підвищення ефективності роботи з науково-популярними текстами. Запропоноване рішення має потенціал для використання в освітніх проєктах, де необхідне швидке отримання стислого змісту великих текстів. Крім того, метод може бути корисним для редакторів, журналістів, дослідників і розробників інформаційних систем, що працюють із великими обсягами текстової інформації.

**Апробація роботи.** Деякі положення і результати роботи були представлені та обговорювались на XVII науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2024 (Київ, 20-22 листопада 2024 р.) та опубліковані у збірнику тез доповідей (С. 104-109).

**Структура та обсяг роботи.** Дослідження складається зі вступу, чотирьох розділів, висновків і додатків.

У вступі описано ситуацію в галузі на сьогодні, яка є підґрунтям для подальшої постановки задачі.

У першому розділі сформульовано проблематику, яке є підставою для дослідження, окреслено актуальність даного дослідження, а також проаналізовано наявні рішення узагальнення великорозмірних текстів, розглянуто їхні переваги та недоліки, а також виділено спільні особливості.

Другий розділ містить опис методів, на основі яких було розроблено комбінований метод, їх недоліки та способи їх вирішення, а також покроковий опис запропонованого методу та обґрунтування метрик для оцінювання ефективності та обраного еталонного методу для порівняння ефективності.

У третьому розділі обґрунтовано обрані засоби для розробки програмного забезпечення, описано архітектуру та особливості реалізації розробленого програмного забезпечення, а також наведено приклад його використання.

У четвертому розділі проаналізовано швидкість роботи запропонованого методу для оцінювання успішності його розробки, а також оглянуто приклад результату роботи методу, після чого визначено подальші потенційні напрямки дослідження.

У висновках узагальнено результати роботи.

Робота виконана на 88 аркушах, містить 3 додатки та посилання на 52 джерела. У роботі наведено 10 рисунків та 7 таблиць.

**Ключові слова:** інженерія програмного забезпечення, екстрактне узагальнення, абстрактне узагальнення, кластеризація, обробка природномовних даних, HDBSCAN, BART, PEGASUS.

## ABSTRACT

**Topic relevance:** As of today, there are a number of solutions in the field of automated text summarization, including both extractive and abstractive types. These methods allow to reduce the size of natural language texts to concise summaries while preserving their essence. However, most modern approaches to text summarization demonstrate high efficiency mainly on short texts, such as news articles. Over time, methods for processing long texts have been developed, but they require significant computational resources and a lot of time to produce a high-quality summary. This is due to both the volume of text and the need to analyze the content, stylistic features, and the logic behind the text. In the case of long popular science texts, which have become very popular in recent decades and usually have a wide range of topics, these approaches cannot provide the proper speed and quality of summarization. Therefore, a relevant task is to develop a summarization method that can effectively work with long popular science texts, while maintaining the coherence, consistency, and coverage of key topics of the input text.

**Object of the research** is the process of automated analysis of natural language data.

**Subject of the research** is methods, techniques, and software tools for summarizing natural language text.

**Research objective** is to increase the speed of the process of summarization of long popular science texts while maintaining such qualitative characteristics of the summarization result as coherence, consistency, and coverage of key topics.

**Scientific novelty.** For the first time a method for summarizing long popular science texts is proposed that creates a coherent and consistent result that covers the key topics of the input text and has at least 1.8 times faster execution speed than the multi-step method of abstract summarization using context-driven chunking.

**Practical value.** The developed method can serve as a basis for further research in the field of natural language processing and for improving the efficiency of working with popular science texts. The proposed solution has the potential to be used in educational projects that require a quick summary of long popular science texts. In addition, the method can be useful for editors, journalists, researchers, and information system developers working with large amounts of textual information.

**Aprobation.** Some findings and results of this research were presented and discussed at the XVIIth scientific conference “Applied mathematics and computing” AMC-2024 (Kyiv, November 20-22, 2024) and published in the proceedings (p. 104-109).

**Structure and content of the thesis.** This thesis consists of an introduction, four chapters, conclusion and appendices.

The introduction describes the current situation in the industry, which is the basis for further formulation of the task.

The first chapter formulates the problem that is the basis for the study, outlines the relevance of this research, and analyzes existing solutions for summarizing long texts, discusses their advantages and disadvantages, and highlights common features.

The second chapter contains a description of the methods on the basis of which the combined method was developed, their shortcomings and ways to solve them, as well as a step-by-step description of the proposed method and justification of the selected metrics for evaluating the effectiveness and the chosen reference method for comparing the effectiveness.

The third chapter describes the selected software development tools, as well as the architecture and implementation features of the developed software, and provides an example of its use.

The fourth chapter analyzes the speed of the proposed method to assess the success of its development, as well as reviews an example of the method's execution result, and finally identifies potential areas of further research.

The conclusion summarizes the results of the research.

The thesis is presented on 88 pages, contains 3 appendices and 52 references, as well as 10 figures and 7 tables.

**Keywords:** software engineering, abstractive summarization, extractive summarization, clustering, natural language processing, HDBSCAN, BART, PEGASUS.

## ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ .....	4
ВСТУП .....	6
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА НАЯВНИХ РІШЕНЬ .....	8
1.1. Поняття, мета та види процесу узагальнення тексту .....	8
1.2. Поняття великорозмірного тексту.....	11
1.3. Поняття науково-популярної літератури.....	12
1.4. Аналіз підходів та методів узагальнення великорозмірного тексту	13
1.5. Аналіз тенденцій щодо використання кордонних обчислень .....	21
1.6. Висновки до розділу .....	24
2. РОЗРОБЛЕННЯ МЕТОДУ УЗАГАЛЬНЕННЯ ВЕЛИКОРОЗМІРНИХ НАУКОВО-ПОПУЛЯРНИХ ТЕКСТІВ .....	27
2.1. Застосування методу екстрактного узагальнення з використанням кластеризації та способи його покращення.....	27
2.2. Застосування методу абстрактного узагальнення з поділом тексту на фрагменти.....	31
2.3. Комбінований метод узагальнення великорозмірних науково- популярних текстів .....	32
2.4. Вибір метрик для визначення кількісних характеристик ефективності пропонованого методу .....	37
2.5. Висновки до розділу .....	44
3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ КОМБІНОВАНОГО МЕТОДУ УЗАГАЛЬНЕННЯ ВЕЛИКОРОЗМІРНИХ НАУКОВО-ПОПУЛЯРНИХ ТЕКСТІВ.....	46
3.1. Обґрунтування вибору засобів реалізації програмного забезпечення .....	46
3.2. Архітектура розробленого програмного забезпечення.....	56
3.3. Особливості реалізації програмного забезпечення .....	58
3.4. Приклад використання розробленого програмного забезпечення ...	64

3.5. Висновки до третього розділу .....	69
4. АНАЛІЗ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ ПРОПОНОВАНОГО МЕТОДУ .....	70
4.1. Критерії оцінювання та аналіз ефективності пропонованого методу.....	70
4.2. Огляд отриманих результатів .....	75
4.3. Напрямки подальшої роботи.....	80
4.4. Висновки до розділу .....	81
ВИСНОВКИ.....	82
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....	84
ДОДАТКИ.....	89

## СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

API – Application Programming Interface – це набір визначених інтерфейсів, протоколів, процедур і структур даних, який дозволяє програмному забезпеченню взаємодіяти з іншими програмами, модулями або сервісами. API визначає контракти взаємодії між клієнтом і постачальником послуг або між компонентами системи.

NLP – Natural Language Processing (обробка природномовних даних) – це міждисциплінарна область, що поєднує лінгвістику, інформатику та штучний інтелект, зосереджуючись на алгоритмах та методах, які дозволяють системам обробляти та моделювати природну мову в її письмовій або усній формі.

N-грама – це послідовність із  $n$  елементів із заданої вибірки, найчастіше послідовність символів, слів або інших одиниць у межах тексту або мовного корпусу.

Токен – це елементарна одиниця тексту, на яку розбивається вхідна інформація для обробки моделлю. Це може бути слово, частина слова, або навіть розділовий знак, залежно від способу токенизації.

UML – це уніфікована мова моделювання, яка використовується для візуалізації, специфікації, конструювання та документування програмних систем.

Семантика – це розділ лінгвістики, що вивчає значення мовних одиниць, таких як слів, фраз або речень, і те, як ці значення формуються, передаються та інтерпретуються.

Модель машинного навчання – це параметризована функція, яка навчається на даних за допомогою тривального алгоритму, щоб виявляти закономірності та робити прогнози або класифікації щодо нових, невідомих даних.

ШІ (штучний інтелект) – це галузь інформатики, що займається створенням систем і програм, які можуть виконувати завдання, що зазвичай

вимагають людського інтелекту, такі як розпізнавання мови, розуміння тексту, ухвалення рішень, навчання та планування.

## ВСТУП

Текстові дані відіграють важливу роль у повсякденному обміні та засвоєнні інформації, особливо в форматі змістовного тексту середнього чи великого розміру. Книги, статті, дослідницькі огляди – всі ці тексти містять значні обсяги інформації, часто викладені на сотнях сторінок. У зв'язку з постійним зростанням кількості таких текстів виникає потреба в ефективному способі ознайомлення з їх змістом без необхідності перегляду всього документа. В таких випадках в нагоді стає технологія автоматичного узагальнення текстів – одна з ключових задач у галузі NLP – обробки природномовних даних.

Сенс автоматичного узагальнення полягає в тому, щоб на основі текстового документу створити короткий виклад його суті, який зберігає основні ідеї та смислове навантаження оригіналу. Це дає змогу користувачеві швидко отримати уявлення про зміст документа, заощаджуючи час та ресурси. Особливо актуальним це є для науково-популярних творів, які за останні кілька десятиліть набули значної популярності та зачасти є текстами великого розміру, що ускладнює їх ручне узагальнення.

Впродовж розвитку галузі обробки природномовних даних було запропоновано безліч методів та підходів узагальнення, що ґрунтуються на екстрактних методах, які вибирають ключові речення з тексту, та абстрактних методах, що формують нові речення на основі розуміння вмісту. Кожен із підходів має свої переваги та недоліки, адже екстрактні методи зазвичай простіші й швидші у виконанні, однак повертають фрагментований і незв'язний результат, а абстрактні методи здатні створювати більш природні й зв'язні підсумки, але потребують значних обчислювальних ресурсів, що робить їх менш придатними для використання на звичайних пристроях користувачів.

Незважаючи на значні досягнення в галузі, досі не існує універсального методу узагальнення, який би забезпечував швидкість обробки, якість результату узагальнення та можливість виконання на персональних пристроях користувачів одночасно. Особливо складною задачею є реалізація такого методу для великорозмірних текстів, які охоплюють широкий спектр тем і потребують більше ресурсів та часу для обробки. Серед інших проблем є надлишкова повторюваність у результаті узагальнення, неповне охоплення основних тем і загальна складність інтеграції такого програмного забезпечення у користувацькі пристрої.

Відтак, актуальним напрямом досліджень є розробка методу узагальнення великорозмірних науково-популярних текстів, який має на меті об'єднати швидкість роботи, повне охоплення ключових тем, зв'язність результату узагальнення і можливість запуску на середньостатистичному користувацькому пристрої.

# 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА НАЯВНИХ РІШЕНЬ

## 1.1. Поняття, мета та види процесу узагальнення тексту

Процес узагальнення тексту – це процес створення короткої версії документа, яка зберігає основну інформацію і смислове навантаження оригінального тексту [1].

Метою створення короткої версії документа (далі – результату процесу узагальнення) є збереження часу читача. Через велику кількість доступних текстових документів читачу стає практично неможливо переглянути та проаналізувати їх на відповідність своїм цілям чи інтересам для подальшого читання, аналізу чи обробки, адже перед читачем постають проблеми перенасичення деталями та нестачі часу. Натомість результат процесу узагальнення, що викладає основні ідеї тексту, дозволяє швидше ознайомитися зі змістом великої кількості текстів.

Метою самого процесу узагальнення є створення короткого тексту, що надасть читачу інформацію про сенс оригінального (далі – вхідного) тексту, при цьому забезпечуючи певні якісні характеристики результату та кількісні характеристики процесу узагальнення. Найважливіша кількісна характеристика процесу узагальнення – швидкість обробки тексту, адже зазвичай читач прагне отримати результат за найкоротший можливий проміжок часу. До якісних характеристик результату узагальнення віднесемо такі властивості, як зв'язність, послідовність та охоплення тем вхідного тексту. В комбінації вони забезпечують читачу цілісний та логічний природномовний текст. Для подальшого використання визначимо наступні вищезгадані якісні характеристики результату узагальнення: зв'язність та послідовність.

Зв'язність тексту – це властивість, яка характеризує структурну цілісність всередині речень та між окремими реченнями або частинами тексту, тобто те, як елементи тексту, зокрема речення, з'єднані між собою через різні механізми мовної організації, що дозволяє забезпечити їхню

взаємодію та плавний перехід. Зв'язність тексту досягається для того, щоб читач міг легко стежити за ходом думки автора і сприймати логічні внутрішньореченнєві та міжреченнєві зв'язки. Вона визначає синтаксичну цілісність, допомагаючи уникнути розірваних або нелогічних переходів між реченнями, та досягається за допомогою наступних механізмів:

- використання сполучників, часток, артиклів та інших мовних засобів для позначення зв'язків між частинами тексту (наприклад послідовність, причина, наслідок, порівняння, контраст, тощо);
- вживання заміників – використання займенників, синонімів, перефразувань, що дозволяє уникати повторів і підтримувати зв'язок між частинами тексту, не порушуючи його логічної послідовності.

Послідовність тексту – це властивість, яка характеризує логічну та змістову цілісність і чітку організацію інформації між реченнями, тобто те, як кожне наступне речення або частина тексту логічно впливає з попереднього і підтримує єдину тему або основну ідею блоку речень. Послідовність тексту досягається для того, щоб читач правильно і послідовно сприймав подану інформацію та не плутався між розглянутими в тексті темами. Таким чином, послідовність тексту забезпечується за допомогою дотримання чіткої логічної структури, де кожне речення тематичного блоку слугує розвитку теми блоку, забезпечуючи зрозумілість та логічність для читача.

Процес узагальнення може виконуватись як вручну, так і за допомогою комп'ютерних систем. У процесі автоматизованого узагальнення зазвичай застосовують попередню обробку тексту, яка включає токенізацію, лематизацію, видалення стоп-слів, тощо. Обидва випадки передбачають аналіз змісту та лексичних елементів тексту з подальшим виокремленням головних смислових одиниць. Автоматизовані

системи узагальнення будуються на основі алгоритмів обробки природної мови, статистичних методів, а також моделей машинного навчання [2].

Розрізняють два основні види процесу узагальнення тексту: екстрактний та абстрактний. Екстрактний процес узагальнення полягає у відборі окремих речень з тексту, які вважаються найбільш інформативними. Відібрані речення при цьому залишаються без змін, тож результат такого виду узагальнення найчастіше не є зв'язним. Послідовність результату та охоплення основних тем залежить від застосованих методів та підходів. Відбір речень базується на аналізі частоти слів, важливості термів, зв'язків між реченнями та інших показників. Часто для цього використовують такі методи, як метрику TF-IDF [3], графові алгоритми, методи класифікації і навіть нейронні мережі. Натомість абстрактний процес узагальнення передбачає генерацію нового тексту, який є результатом інтерпретації, узагальнення і переформулювання змісту документа, за допомогою моделей абстрактного узагальнення, що ґрунтуються на нейронних мережах та трансформерах [4]. Таким чином результат абстрактного узагальнення найчастіше є зв'язним та охоплює основні теми тексту. Послідовність результату залежить від допоміжних підходів та тренування моделей машинного навчання.

Досі не існує універсального методу для узагальнення текстів, адже процес узагальнення тексту є складною задачею, що вимагає врахування не лише лексичних особливостей, а й семантичних зв'язків, логічної структури та стилістики тексту. Окрім того, наявні базові екстрактні та абстрактні методи мають недоліки, що унеможливають ефективну обробку навіть окремих жанрів текстів. Сучасні дослідження спрямовані на поєднання обох типів узагальнення шляхом створення комбінованих методів, що спочатку оброблюють текст екстрактно, а далі абстрактно, використовуючи переваги обох типів узагальнення.

## 1.2. Поняття великорозмірного тексту

У процесі абстрактного узагальнення важливу роль відіграє обсяг вхідного тексту. Попри те, що абстрактні моделі демонструють набагато вищу якість результатів обробки тексту порівняно з екстрактними методами, вони повільні у виконанні та потребують набагато більше обчислювальних ресурсів через їх квадратичну залежність від довжини вхідного тексту [5]. Тому сучасні моделі, зокрема ті, що базуються на трансформерах, мають обмеження на довжину оброблюваного тексту. Це обмеження визначається у токенах – базових одиницях, на які текст розбивається перед обробкою.

Для більшості актуальних моделей обмеження довжини вхідної послідовності становить 1024 токени, що є рівним близько 870 словам. Такої кількості може вистачати для коротких документів, таких як новинні замітки чи окремі параграфи, однак цього недостатньо для обробки великих статей чи книг. Якщо текст перевищує встановлене обмеження за кількістю токенів, для його повної обробки моделлю абстрактного узагальнення текст потрібно буде розбити на частини й робити виклик моделі  $n$  разів, що може суттєво збільшити загальний час роботи системи. Коли текст у понад 50 разів перевищує допустиму довжину вхідної послідовності (тобто містить понад 51200 токенів), час обробки стає неприйнятно великим. У зв'язку з цим доцільно вважати великорозмірними такі тексти, довжина яких перевищує 51200 токенів. Такий поріг був визначений з точки зору ефективності обробки та доцільності ресурсозатрат. Робота з великорозмірними текстами вимагає спеціальних підходів до попередньої обробки, включаючи сегментацію тексту та виявлення ключових тематичних блоків, а також комбінування обох видів процесу узагальнення, тож при створенні методів узагальнення великорозмірного тексту необхідно враховувати обсяг вхідних даних та адаптувати методи обробки, щоб забезпечити прийнятну швидкість обробки та якість результатів.

### 1.3. Поняття науково-популярної літератури

Науково-популярна література – це вид літератури, що доступно та зрозуміло розповідає про наукові знання широким верствам населення, які не мають спеціальної фахової підготовки та не пов'язані з певною науковою галуззю [6]. Тексти такого виду виступають посередником між академічною наукою та масовим читачем, поєднуючи достовірність і точність наукової інформації з виразністю, образністю і простотою викладу. Основною метою науково-популярного тексту є пояснення складних понять простою мовою, формування загальної ерудиції, популяризація наукових досягнень та стимулювання інтересу до науки. Цей тип тексту часто використовується у енциклопедіях, документальних передачах, підручниках для широкого загалу, а також у книгах, орієнтованих на нефхівців. До характерних рис науково-популярного тексту належать:

- доступність викладу – використання простої, часто розмовної мови замість спеціалізованої термінології, або пояснення останньої у зрозумілій формі;
- інформативність – текст містить достовірну наукову інформацію, що базується на фактах;
- логічна структура – наявність чітко вираженого вступу, основної частини та висновку;
- пояснювально-роз'яснювальна функція – автор прагне не лише подати інформацію, а й пояснити її значення, взаємозв'язки та прикладне застосування;
- наявність прикладів, аналогій, метафор – для полегшення сприйняття складного матеріалу.

Завдяки цим характеристикам, науково-популярні тексти є вдалими кандидатами для автоматизованого узагальнення. По-перше, їхня структурована побудова, а саме наявність послідовного викладу фактів та висновків, відповідає очікуванням щодо логічного розташування ключової інформації, завдяки чому не потрібно проводити комплексний аналіз

структури тексту. По-друге, через відсутність рідковживаної спеціалізованої термінології та просту граматичну конструкцію, такі тексти знижують ризик неправильного перефразування, додавання неправдивої інформації або втрати сенсу під час процесу абстрактного узагальнення, а також не мають потреби в тренуванні моделі абстрактного узагальнення на спеціалізованих текстах. Окрім того, науково-популярні тексти зазвичай зосереджені на поясненні кількох центральних понять, що спрощує виділення головних тематичних блоків та ключових тверджень, які повинні бути збережені у результаті узагальнення. У сукупності ці фактори роблять науково-популярні тексти одним з найбільш підходящих кандидатів для розробки покращеного методу узагальнення великорозмірних текстів.

Серед великорозмірних науково-популярних текстів, більшість з яких є книгами, можна виділити наступні найбільш популярні категорії:

- астрономія та космологія;
- фізика та математика;
- біологія та генетика;
- історія та еволюція;
- медицина та нейронауки;
- психологія та когнітивістика;
- інформатика та цифрові технології;
- землезнавство та екологія;
- хімія та матеріалознавство.

Зразки текстів даних категорій в подальшому будуть використані для оцінювання ефективності розробленого методу.

#### **1.4. Аналіз підходів та методів узагальнення великорозмірного тексту**

Узагальнення великорозмірних текстів є важким та нетривіальним завданням, адже результатом узагальнення має бути стислий, зв'язний та послідовний виклад суті тексту, що повністю охоплює значні тематичні

блоки тексту, в той час як процес узагальнення має мати високу швидкість обробки тексту. Базові методи екстрактного узагальнення демонструють високу швидкість обробки, однак вони не здатні створити зв'язний послідовний результат, тоді як моделі абстрактного узагальнення створюють зв'язний та послідовний результат, однак мають обмеження на обсяг вхідного тексту та досить низьку швидкість обробки. У зв'язку з цим були запропоновані різноманітні стратегії та способи, що дозволяють масштабувати процес узагальнення до великих обсягів текстових даних. Нижче наведено список найпоширеніших методів узагальнення великорозмірних текстів з докладним описом їх переваг та недоліків.

1. *Екстрактне узагальнення за допомогою базових методів.*

Виконується шляхом обробки тексту в один прохід методами на кшталт TextRank [7] або LexRank [8]. Першим кроком будується матриця подібності речень шляхом обчислення міри косинус-подібності векторних відображень між кожною парою речень. Далі будується граф ранжування речень та обчислюються оцінки речень за алгоритмом PageRank [9]. Останнім кроком обирається  $k$  найважливіших речень.

Переваги:

- немає обмежень на обсяг вхідного тексту – на відміну від абстрактних моделей базові методи використовують прості алгоритми, що не потребують багато системних ресурсів, як наслідок немає обмеження на обсяг текстових даних;
- висока швидкість обробки – вищезгадані алгоритми використовують прості математичні операції, що сприяє високій швидкості виконання.

#### Недоліки:

- незв'язний результат – висока ймовірність нелогічних або розірваних переходів між реченнями, оскільки процес екстрактного узагальнення лише вибирає речення без їх зміни;
- непослідовний результат – висока ймовірність невідповідності порядку вибраних речень початковому логічному ланцюжку, висока ймовірність змішування речень з різних тематичних блоків;
- неповне охоплення тематичних блоків – через останній крок алгоритму – обрання  $k$  найважливіших речень – не всі тематичні блоки можуть потрапити в кінцеву вибірку.

2. *Абстрактне узагальнення урізаного тексту.* Виконується шляхом виділення блоку тексту на початку документа та узагальнення його абстрактною моделлю [10]. Було розроблене для виконання узагальнення новинних статей, адже в текстах такого формату основна інформація в основному зосереджена на початку тексту, щоб зацікавити читача, а після цього новина описана в подробицях.

#### Переваги:

- немає обмежень на обсяг вхідного тексту – оскільки для узагальнення береться лише перші 1024 токени, а решта тексту ігнорується;
- висока швидкість обробки – оскільки відбувається лише один виклик моделі абстрактного узагальнення;
- зв'язний результат – оскільки в процесі абстрактного узагальнення забезпечується взаємодія та плавний перехід між реченнями.

Недоліки:

- непослідовний результат – частина початкового логічного ланцюжка може бути втрачена через урізання тексту;
- неповне охоплення тематичних блоків – оскільки більшість тексту ігнорується.

3. *Комбінований метод з застосуванням екстрактного та абстрактного узагальнення.* Першим кроком виконується процес екстрактного узагальнення (наприклад методом TextRank) для зменшення розміру початкового тексту шляхом вибору найважливіших речень. Далі вибірка речень проходить процес узагальнення абстрактною моделлю [11].

Переваги:

- немає обмежень на обсяг вхідного тексту – оскільки першим виконується екстрактне узагальнення, що не має обмежень, а абстрактне узагальнення виконується на отриманій вибірці розміром до 1024 токенів;
- висока швидкість обробки – оскільки у кроку екстрактного узагальнення висока швидкість обробки, а абстрактна модель викликається лише один раз;
- зв'язний результат – оскільки на останньому кроці викликається модель абстрактного узагальнення.

Недоліки:

- неповне охоплення тематичних блоків – через проблему з охопленням у першого кроку – процесу екстрактного узагальнення;
- непослідовний результат – низька ймовірність того, що модель абстрактного узагальнення виправить непослідовну вибірку – результат першого кроку.

4. *Екстрактне узагальнення з застосуванням кластеризації.* Для речень обчислюються векторні відображення, що групуються у кластери, які відображають тематичні блоки тексту. З кожного такого кластера обирається  $k$  найбільш значущих речень [12].

Переваги:

- висока швидкість обробки – методи обчислення векторних відображень та кластеризації мають високу швидкодію;
- повне охоплення тематичних блоків – при кластеризації кожен кластер відображає певний значущий тематичний блок, тоді як речення-викиди не належать до конкретної теми;
- немає обмежень на обсяг вхідного тексту – методи кластеризації здатні ефективно обробляти набагато більші обсяги даних, ніж кількість речень у середньостатистичному великорозмірному тексті (близько 6000 речень).

Недоліки:

- незв'язний результат – наявність нелогічних або розірваних переходів між реченнями через обрання речень без їх зміни та підлаштування;
- непослідовний результат – якщо обирати речення в порядку значущості без врахування їх послідовності у вхідному тексті, результат з високою ймовірністю втратить логічний ланцюжок зв'язків між реченнями.

5. *Багатокрокове абстрактне узагальнення з застосуванням поділу тексту на фрагменти.* Першим кроком текст ділиться на фрагменти за одним із методів фрагментації: фрагментація фіксованого розміру (наприклад кожен фрагмент має довжину 2000 символів незалежно від меж речень), фрагментація

визначена вмістом (наприклад кожен фрагмент має довжину не більше 3000 символів і є набором необрізаних речень) або фрагментація визначена контекстом та тематикою (кожен фрагмент виділений так, що він має власну тему і не перетинається з іншими фрагментами, відповідно також будучи набором необрізаних речень) [13]. Далі кожен з фрагментів узагальнюється моделлю абстрактного узагальнення, результати з'єднуються в єдиний текст [14]. Якщо фрагмент містить більше токенів, ніж може обробити модель, відбувається повторна фрагментація. Якщо кінцевий результат недостатньо стислий, процес починається з початку, де вхідний текст – результат попередньої ітерації методу.

Переваги:

- повне охоплення тематичних блоків тексту – у випадку фрагментації визначеної контекстом кожен тематичний блок узагальнюється абстрактною моделлю незалежно від інших блоків та включається в проміжні та кінцевий результати;
- зв'язний результат – внаслідок застосування моделі абстрактного узагальнення;
- послідовний результат – внаслідок застосування узагальнення до окремих тематичних блоків та з'єднання у порядку, визначеному у вхідному тексті;
- немає обмеження на розмір вхідного тексту – через застосування фрагментації тексту.

Недоліки:

- дуже низька швидкість обробки – через обробку кожного фрагмента моделлю абстрактного узагальнення, що сама по собі має низьку швидкість обробки тексту.

6. *Абстрактне узагальнення з застосуванням моделі лонгформера.* Лонгформери – це моделі машинного навчання з модифікованим механізмом уваги, що дозволяє змінити квадратичну залежність необхідної пам'яті від довжини вхідного тексту на лінійну залежність [5]. Це означає, що моделі лонгформери можуть обробляти набагато довший текст, ніж звичайні моделі з повним механізмом уваги, за умови ідентичного використання пам'яті. Зазвичай моделі лонгформери можуть обробляти текст обсягом до 16000 токенів. Однак для великорозмірних текстів цього все ще недостатньо, тому такі моделі все ще мають застосовуватися як складова комбінованого методу.

Переваги:

- висока швидкість обробки тексту – набагато вища ніж в звичайної моделі, адже за один прохід можна обробити приблизно в 15 разів довший текст;
- зв'язність результату – як і в стандартної моделі абстрактного узагальнення результат буде зв'язний.

Недоліки:

- обмеження на обсяг вхідного тексту – присутнє як і в стандартних моделей машинного навчання;
- непослідовний результат – через модифікований механізм уваги частина важливих речень може не бути включена в результат, що призведе до втрати логічного ланцюжка;
- неповне охоплення тематичних блоків тексту – наразі немає великої кількості датасетів з текстами довжиною близько 16000 токенів та їх узагальненнями відповідної довжини, що унеможлиблює правильне тренування моделі і як наслідок не всі теми вхідного тексту потрапляють в результат узагальнення.

Для зручності огляду переваги та недоліки методів винесені у таблицю 1.1.

Таблиця 1.1

Переваги та недоліки методів узагальнення великорозмірних текстів

	Висока швидкість обробки	Немає обмежень на обсяг вхідного тексту	Зв'язний результат	Послідовний результат	Повне охоплення тематичних блоків тексту
Екстрактне узагальнення за допомогою базових методів	Так	Так	Ні	Ні	Ні
Абстрактне узагальнення урізаного тексту	Так	Так	Так	Ні	Ні
Комбінований метод з застосуванням екстрактного та абстрактного узагальнення	Так	Так	Так	Ні	Ні
Екстрактне узагальнення з застосуванням кластеризації	Так	Так	Ні	Ні	Так

Продовження табл. 1.1

Багатокрокове абстрактне узагальнення з застосуванням поділу тексту на фрагменти	Ні	Так	Так	Так	Так
Абстрактне узагальнення з застосуванням моделі лонгформера	Так	Ні	Так	Ні	Ні

Як бачимо, кожен з методів узагальнення великорозмірних текстів має свої переваги та недоліки. Метод багатокрокового абстрактного узагальнення з застосуванням поділу тексту на фрагменти має критичний недолік – низьку швидкість обробки, однак він найбільшою мірою охоплює розглянуті якісні характеристики результату узагальнення, а саме зв'язність та послідовність, а також охопленням всіх тематичних блоків тексту. Саме тому розробка нового методу узагальнення великорозмірних текстів орієнтована на досягнення таких самих якісних характеристик результату узагальнення при збільшеній швидкості виконання процесу узагальнення.

### **1.5. Аналіз тенденцій щодо використання кордонних обчислень**

З розвитком технологій машинного навчання та штучного інтелекту все більше і більше людей починають користуватися онлайн засобами на кшталт ChatGPT [15], тож можна припустити, що в недалекому майбутньому більшість людей з доступом до обчислювальної техніки будуть використовувати моделі штучного інтелекту щодня. Це спричинить надмірне навантаження на мережу Інтернет та хмарну інфраструктуру таких сервісів, тож теперішні дослідження зосереджені на ефективних методах та підходах обчислень, здатних забезпечувати низьку затримку, зменшене

навантаження на мережу і хмарні сервіси та конфіденційність обробки. Одним із таких підходів є кордонні обчислення – парадигма, згідно з якою обробка даних здійснюється не в хмарних дата-центрах, а безпосередньо на периферії мережі, тобто на кордонних пристроях, до яких серед іншого належать персональні комп'ютери, смартфони та планшети [16].

Провідні технологічні компанії світу вже активно застосовують концепцію кордонних обчислень у своїх продуктах, адаптуючи моделі штучного інтелекту та машинного навчання до локального виконання:

- компанія Apple інтегрувала фреймворк Core ML у свої операційні системи, що дозволяє розпізнавання облич, мови, зображень, а також обробку тексту безпосередньо на iPhone або MacBook [17];
- компанія Samsung у своїх флагманських смартфонах реалізувала локальну обробку фото, відео, голосу, тощо за допомогою штучного інтелекту без запитів до серверів [18];
- компанія Amazon просуває платформу AWS Greengrass, яка дозволяє створювати, тестувати та застосовувати машинне навчання на пристроях IoT за відсутності доступу до хмари [19].

Використання кордонних обчислень замість хмарних обчислень у завданні узагальнення великорозмірних текстів забезпечує такі переваги, як:

- зменшення часу виконання та навантаження на мережу – при локальній обробці не потрібно надсилати файл з текстом на віддалений сервер, що зменшує використання мережевого трафіку та загальний час обробки;
- відсутність навантаження на хмарні сервіси – обробка відбувається на пристроях користувачів, що дає змогу усунути безліч запитів до віддалених серверів;
- зменшення вартості обслуговування – за відсутності необхідності тримати у стані готовності велику кількість

ресурсозатратних обчислювальних центрів для швидкої роботи хмарних сервісів;

- незалежність від стану мережі – при локальному виконанні текст буде оброблено навіть в умовах обмеженої пропускної здатності або нестабільного з'єднання;
- підвищення конфіденційності – дані не виходять за межі пристрою користувача, що знижує ризик перехоплення чи несанкціонованого доступу.

З урахуванням цих чинників, обробка великорозмірних текстів локально без передачі в хмару є логічним та ефективним рішенням, яке дозволить зменшити витрати ресурсів та навантаження на хмарні сервіси, забезпечити захист персональних даних і досягти більшої автономності роботи. Однак така обробка має суттєві обмеження, пов'язані передусім із апаратними характеристиками пристроїв користувачів, більшість з яких не мають потужних графічних процесорів або необхідних для них драйверів та програмних адаптерів, що дозволяють ефективно паралелізувати виконання коду моделей машинного навчання, які в випадку виконання за допомогою лише центрального процесора мають погіршену ефективність та відповідно нижчу швидкість виконання. Тому для забезпечення комфортного використання на користувацьких пристроях розроблений метод узагальнення великорозмірного тексту має реалізовувати наступні вимоги щодо кількісних показників та технічних можливостей:

- має бути збільшена швидкість обробки тексту в порівнянні з методом багатокрокового абстрактного узагальнення з застосуванням поділу тексту на фрагменти – для прискорення отримання користувачем результату узагальнення з ідентичними якісними характеристиками;
- має бути можливість виконання методу на середньостатистичному користувацькому пристрої за оцінюванням станом на 2025 рік – не менше 12 ГБ оперативної

пам'яті, центральний процесор з не менш ніж 6 ядрами та базовою частотою не менше 2.6 ГГц;

- перевірка та оцінювання ефективності методу мають бути проведені без використання окремо виділеного графічного процесора – адже немає гарантії, що у всіх користувачів будуть відповідні можливості застосування паралельних обчислень.

Кордонні обчислення є відповіддю на виклики сучасної цифрової епохи, пов'язані з масштабом, швидкістю надходження та кількістю запитів на обробку даних. У контексті узагальнення великорозмірних текстів цей підхід дозволяє розв'язати низку важливих задач – від зменшення мережевого навантаження до своєчасного забезпечення користувачів результатами обробки текстів. Водночас даний підхід вимагає розробки методу на основі новітніх комбінованих підходів та використання ефективних та швидких алгоритмів та моделей машинного навчання для швидкого та ефективного виконання навіть на малопотужних пристроях.

## **1.6. Висновки до розділу**

У даному розділі визначено поняття процесу узагальнення тексту як процес скорочення вхідного тексту зі збереженням його основної ідеї. Також було визначено поняття мети цього процесу як створення скороченої версії тексту при забезпеченні певних якісних характеристик результату та кількісних характеристик процесу узагальнення. Метою створення узагальнення є збереження часу читача при потребі обробити велику кількість текстів. Окрім того було визначено такі якісні характеристики результату узагальнення, як зв'язність та послідовність.

Було надано загальну характеристику процесу узагальнення та розглянуто його два основні види: екстрактне та абстрактне узагальнення. Також було визначено необхідність подальших досліджень у сфері узагальнення природномовних текстів та актуальність розробок з використанням комбінації обох видів процесу узагальнення.

Було визначено поняття великорозмірного тексту як текст, довжина якого перевищує 51200 токенів. Даний поріг був визначений з точки зору ефективності обробки та доцільності ресурсозатрат, що свідчить про те, що процес узагальнення великорозмірного тексту вимагає спеціальних підходів до попередньої обробки, комбінування обох видів процесу узагальнення та адаптації методів обробки, щоб забезпечити прийнятну швидкість обробки та якість результатів.

Також було визначено поняття науково-популярної літератури як текст, що доступно та зрозуміло розповідає про наукові знання широкій аудиторії. Такі характеристики, як доступність викладу, інформативність та логічна структура, роблять науково-популярні тексти одним з найбільш підходящих кандидатів для розробки покращеного методу автоматизованого узагальнення великорозмірних текстів.

Основна частина розділу присвячена огляду наявних методів узагальнення великорозмірних текстів та їх переваг та недоліків. Було розглянуто наступні методи:

- екстрактне узагальнення за допомогою базових методів;
- абстрактне узагальнення урізаного тексту;
- комбінований метод з застосуванням екстрактного та абстрактного узагальнення;
- екстрактне узагальнення з застосуванням кластеризації;
- багатокрокове абстрактне узагальнення з застосуванням поділу тексту на фрагменти;
- абстрактне узагальнення з застосуванням моделі лонгформера.

Було визначено, що метод багатокрокового абстрактного узагальнення з застосуванням поділу тексту на фрагменти найбільшою мірою охоплює розглянуті якісні характеристики результату узагальнення, а саме зв'язність та послідовність, а також охопленням всіх тематичних блоків тексту. Тому розробка нового методу узагальнення великорозмірних текстів орієнтована на досягнення таких самих якісних характеристик

результату узагальнення при збільшеній швидкості виконання процесу узагальнення.

Наостанок було розглянуто тенденції щодо використання кордонних обчислень та локального виклику моделей машинного навчання провідними ІТ-компаніями світу. Кордонні обчислення забезпечують значне зниження навантаження з хмарних сервісів, однак вимагають використання швидких та ефективних методів для уможливлення локального виконання.

Таким чином з вищевказаного можна зробити наступні висновки:

- завдання узагальнення великорозмірних текстів є актуальним, нетривіальним та не повністю вирішеним;
- науково-популярна література найбільш підходяща для процесу узагальнення;
- потребує розробки метод узагальнення, що поєднує основні якісні характеристики результату узагальнення для сприйняття читачем та швидкість і ефективність процесу узагальнення для забезпечення можливості локального виконання.

## **2. РОЗРОБЛЕННЯ МЕТОДУ УЗАГАЛЬНЕННЯ ВЕЛИКОРОЗМІРНИХ НАУКОВО-ПОПУЛЯРНИХ ТЕКСТІВ**

### **2.1. Застосування методу екстрактного узагальнення з використанням кластеризації та способи його покращення**

За основу першого кроку пропонованого методу було вирішено взяти метод екстрактного узагальнення з застосуванням кластеризації, адже при аналізі наявних методів та підходів узагальнення великорозмірних текстів було визначено, що він забезпечує високу швидкість обробки, охоплення всіх тем вхідного тексту та не має обмежень на розмір вхідного тексту.

Використання кластеризації є добре перевіреним способом екстрактного узагальнення, про що свідчать безліч наукових праць. Серед останніх робіт можна виділити “Модель автоматичного узагальнення заснована на алгоритмі кластеризації” [20]. Автори застосовують власну модель розроблену на основі BERT [21] для обчислення векторних відображень речень. Далі за допомогою алгоритму k-means [22] проводять кластеризацію речень використовуючи їх векторні відображення, що результує в множину наборів речень, де речення в кожному наборі належать до різних груп в семантичному просторі. Далі автори обирають кілька найбільш значущих речень з кожного набору (кластера), що забезпечує досягнення мети їх дослідження – зменшення семантичної надлишковості в результаті узагальнення, адже таким чином в результуючому тексті знаходиться мінімальна кількість речень, що повторюють одну і ту саму думку. Однак у даного методу присутні наступні проблеми:

- Використання алгоритму k-means не є оптимальним для даної задачі через проблеми пов’язані з логікою алгоритму. По-перше, завідомо невідома кількість кластерів (кількість тематичних блоків), яка має бути задана при ініціалізації алгоритму. При заданні неправильної кількості кластерів в процесі кластеризації будуть неправильно визначені наявні структури даних [23]. По

друге, алгоритм не може правильно обробити структури даних, відмінні від сферичних та ізотропних, тобто видовжені, неправильної форми чи різного розміру кластери [24]. Оскільки невідомо, яким чином векторні відображення сформуують структуру даних для певного тематичного блоку в семантичному просторі, не варто покладатися на алгоритм кластеризації, що не здатний виділяти кластери складної форми.

- В контексті застосування першим кроком комбінованого методу через обрання  $k$  найбільш значущих речень не буде досягатися якісна характеристика “послідовність”. Оскільки речення обираються спираючись лише на значущість без врахування їх початкового порядку, існує висока ймовірність втратити логічний ланцюжок, що був у вхідному тексті.
- При обранні  $k$  найбільш значущих речень з кожного кластера виникає проблема неповного або хибного представлення тематичного блоку (кластера), адже найбільш значущі речення (найближчі до центроїда у випадку  $k$ -means) є “золотою серединою” кластера та у певних випадках не є найбільш інформативними. Візьмемо до прикладу елементи кластера, виділеного під час експерименту над кластеризацією великорозмірного тексту, що наведені в таблиці 2.1. Ймовірність приналежності речення до кластера може знаходитися в діапазоні від 0 до 1, де 1 це 100% ймовірність приналежності. Як бачимо, найбільш значущі речення за визначенням кластеризації, особливо ті, в яких немає конкретного імені чи року, не несуть корисну інформацію, тоді як речення з ймовірністю 0,85 дає зрозуміти, що мова йде про короля Англії Едварда Першого, королеву Елеонору Кастильську та їхніх дітей. Таким чином можна зробити

висновок, що обрання лише кількох найзначущіших речень не забезпечує надійного представлення тематичного блоку.

Таблиця 2.1

Речення – елементи кластера та значення ймовірності їх приналежності

Ймовірність приналежності	Речення
1	An anonymous daughter, born in 1255, died at birth.
1	A daughter, Catherine, died either at age one or age three.
0,99	A daughter, Joan, died at six months.
1	A daughter, Eleanor, died at age twenty-nine.
1	An anonymous daughter died at five months.
1	A daughter, Joan, died at age thirty-five.
1	A daughter, Margaret, died at age fifty-eight.
1	A daughter, Berengeria, died at age two.
1	An anonymous daughter died shortly after birth.
1	A daughter, Mary, died at age fifty-three.
1	An anonymous son died shortly after birth.
1	A daughter, Elizabeth, died at age thirty-four.
0,85	On average, Edward and Eleanor lost a child every three years, ten children one after another.

Окремо виділимо проблему, наявну в багатьох інших роботах, а саме спосіб отримання векторних відображень для речень. Якість кластеризації напряму залежить від якості векторних відображень, адже вони мають відображати контекст слів та контекст їх порядку у реченні. При використанні простих підходів на кшталт побудови векторних відображень на основі TF-IDF, що ніяк не враховує контекст слів та їх порядку, векторні

відображення можуть хибно представляти речення, що спричинить неправильну кластеризацію. Для наочності представимо деякі англійські омоніми як один з найбільш впливових на контекст факторів з відповідними українськими значеннями у таблиці 2.2. Очевидно, що контекстна проблема не обмежується омонімами, а також включає фразеологізми, множинні значення, метафори, заперечення, модальність та інше. Таким чином, для обчислення векторних відображень необхідне використання засобів, здатних враховувати контекст.

Таблиця 2.2

Англійські омоніми та їх відповідники в українській мові

Англійський омонім	Українські значення
bank	банк, берег водойми, кренитися, схилитися
current	поточний, течія, струм
match	спортивний матч, сірник, підходити
right	право (напрямок), право (людини), правильний
rock	скеля, хитатися
saw	пила, прислів'я, побачив
well	колодязь, добре, набагато

Отже, для застосування підходу екстрактного узагальнення з застосуванням кластеризації в межах комбінованого методу узагальнення великорозмірних науково-популярних текстів необхідно забезпечити наступні вимоги:

- використання методу обчислення векторних відображень, що враховує контекст слів та їх порядок в реченні;

- використання методу кластеризації, що не залежить від початкового задання кількості кластерів та здатний обробляти структури даних різної форми та розміру;
- обрання всіх речень з кластера, що мають ймовірність приналежності до кластера більшу або рівну 0.8, для запобігання неповного або хибного представлення тематичного блоку, та у послідовності, визначеній вхідним текстом, для забезпечення якісної характеристики результату узагальнення “послідовність”.

## **2.2. Застосування методу абстрактного узагальнення з поділом тексту на фрагменти**

Отримавши набори речень, що відображають тематичні блоки вхідного тексту після кроку екстрактного узагальнення з використанням кластеризації, маємо наступні проблеми:

- загальний обсяг речень в наборах надто великий для стислого узагальнення – адже замість  $k$  найбільш значущих речень було обрано всі речення, що мають ймовірність приналежності до кластера 0.8 та більше;
- результат кроку екстрактного узагальнення не є зв’язним.

Оскільки необхідне подальше узагальнення, що забезпечить зв’язність результату, доцільно застосувати модель абстрактного узагальнення. Однак через наявність обмеження на обсяг вхідного тексту в таких моделях необхідно проводити додаткове розбиття тексту на фрагменти, що підрозуміває застосування методу абстрактного узагальнення з поділом тексту на фрагменти. Даний метод є загальноживаним способом абстрактного узагальнення текстів, обсяг яких перевищує можливості моделей, про що свідчать безліч наукових праць. Серед останніх робіт можна виділити “Процес абстрактного узагальнення документів англійською мовою з використанням моделі BART та методу

фрагментації” [14]. Автори застосовують попередню обробку тексту та фрагментацію фіксованого розміру перед тим, як обробити кожен фрагмент за допомогою моделі BART.

Для поліпшення даного підходу пропонується застосовувати фрагментацію визначену вмістом замість фрагментації фіксованого розміру. Це дозволить зберегти повний контекст індивідуальних речень, адже на відміну від фрагментації фіксованого розміру не буде можливості поділу речення на дві частини та попадання різних частин у різні фрагменти. Таким чином модель абстрактного узагальнення отримає на вхід множину цілісних речень та забезпечить зв'язний та інформативний результат. Натомість застосування фрагментації визначеної контекстом не є доцільним, адже фрагментація проводиться для набору речень, що є результатом кластеризації і вже має спільний контекст.

### **2.3. Комбінований метод узагальнення великорозмірних науково-популярних текстів**

З врахуванням вищенаведеного, нижче покроково описаний запропонований комбінований метод узагальнення великорозмірних науково-популярних текстів [25].

1. Попередня обробка тексту – приведення тексту у вигляд одного рядка шляхом заміни переходів на новий рядок, табуляції та послідовностей пробілів одним пробілом. Окрім того відбувається знешумлення шляхом видалення спецсимволів, що не несуть інформаційного навантаження.
2. Токенізація тексту – розбиття тексту на речення за допомогою знаків пунктуації. Речення, що не проходять поріг довжини в 30 символів, ігноруються через низьку ймовірність їхньої важливості.
3. Обчислення векторних відображень – для кожного речення обчислюється вектор, що відображає речення в семантичному

просторі. Для цього завдання було використано модель Universal Sentence Encoder [26], а саме варіант, розроблений на базі трансформера [27], для забезпечення врахування контексту слів та їх порядку. Модель повертає 512 вимірний нормалізований вектор для кожного обробленого речення. Під час розробки експериментальним чином було визначено, що при передачі в модель великої кількості речень відбувається переповнення оперативної пам'яті пристрою, що призводить до передчасного завершення роботи коду з помилкою. Оскільки метод розроблюється для виконання на користувацьких пристроях з обмеженими обчислювальними потужностями, для обчислення векторних відображень речення передаються в модель блоками по 1000 речень. Такий спосіб запобігає переповненню пам'яті та має ідентичний час виконання в порівнянні із стандартним способом.

4. Кластеризація векторних відображень – векторні відображення групуються за темами, визначеними їхнім контекстом в семантичному просторі. Зважаючи на визначені вище вимоги до застосування кластеризації для екстрактного узагальнення, був обраний алгоритм HDBSCAN [28], адже він має наступні властивості:

- немає необхідності задавати кількість кластерів при ініціалізації алгоритму;
- набагато краща обробка структур даних неправильної форми та різних розмірів, а також точок-викидів порівняно з іншими алгоритмами кластеризації [29];
- один з найбільш ефективних та швидких методів кластеризації [30].

У результаті кластеризації кожному векторному відображенню, а відповідно і реченню, присвоюється номер кластера та число від 0 до 1 – ймовірність приналежності до кластера.

5. Обрання речень для абстрактного узагальнення – з кожного кластера обираються всі речення, ймовірність приналежності до кластера яких знаходиться в діапазоні від 0,8 до 1 включно. Таким чином виключається можливість неповного або хибного представлення тематичного блоку. Речення зберігаються у послідовності, визначеній вхідним текстом, для збереження їх логічної послідовності.
6. Застосування моделі абстрактного узагальнення – для кожного набору обраних речень перевіряється загальна кількість токенів, що залежить від обраної моделі. Якщо кількість токенів менша за встановлене обмеження моделі, весь набір речень обробляється одним викликом моделі. Якщо кількість токенів більша за обмеження, застосовується фрагментація визначена вмістом. Набір речень ділиться на приблизно рівні частини таким чином, щоб кожна частина була не більша за встановлене обмеження моделі. Далі кожна частина обробляється моделлю. Алгоритм абстрактного узагальнення з застосуванням фрагментації визначеної вмістом зображений на рис. 2.2. Для забезпечення різносторонності та об'єктивності подальшої перевірки ефективності методу за моделі абстрактного узагальнення було обрано BART [31] та PEGASUS [32]. Архітектура обох моделей базується на трансформерах, модель PEGASUS розроблена на основі моделі BART. Ключова різниця між моделями полягає у способі їх тренування та у швидкості їх роботи. Тренування BART проводиться шляхом заміни послідовності слів маскувальним токеном, тоді як для PEGASUS маскувальними токенами замінюються як

послідовності слів, так і цілі речення. Загалом така методика має покращити результати роботи PEGASUS, однак відповідно також збільшує тривалість обробки порівняно з BART.

7. Сортування результатів – отримані результати обробки моделі сортуються в порядку зростання середнього значення позицій речень узагальненого набору у вхідному тексті.
8. Конкатенація результатів – відсортовані результати узагальнення з'єднуються у кінцевий результат.

Загальна схема запропонованого методу наведена на рис. 2.1.



Рис. 2.1. Схема етапів комбінованого методу узагальнення великорозмірних науково-популярних текстів

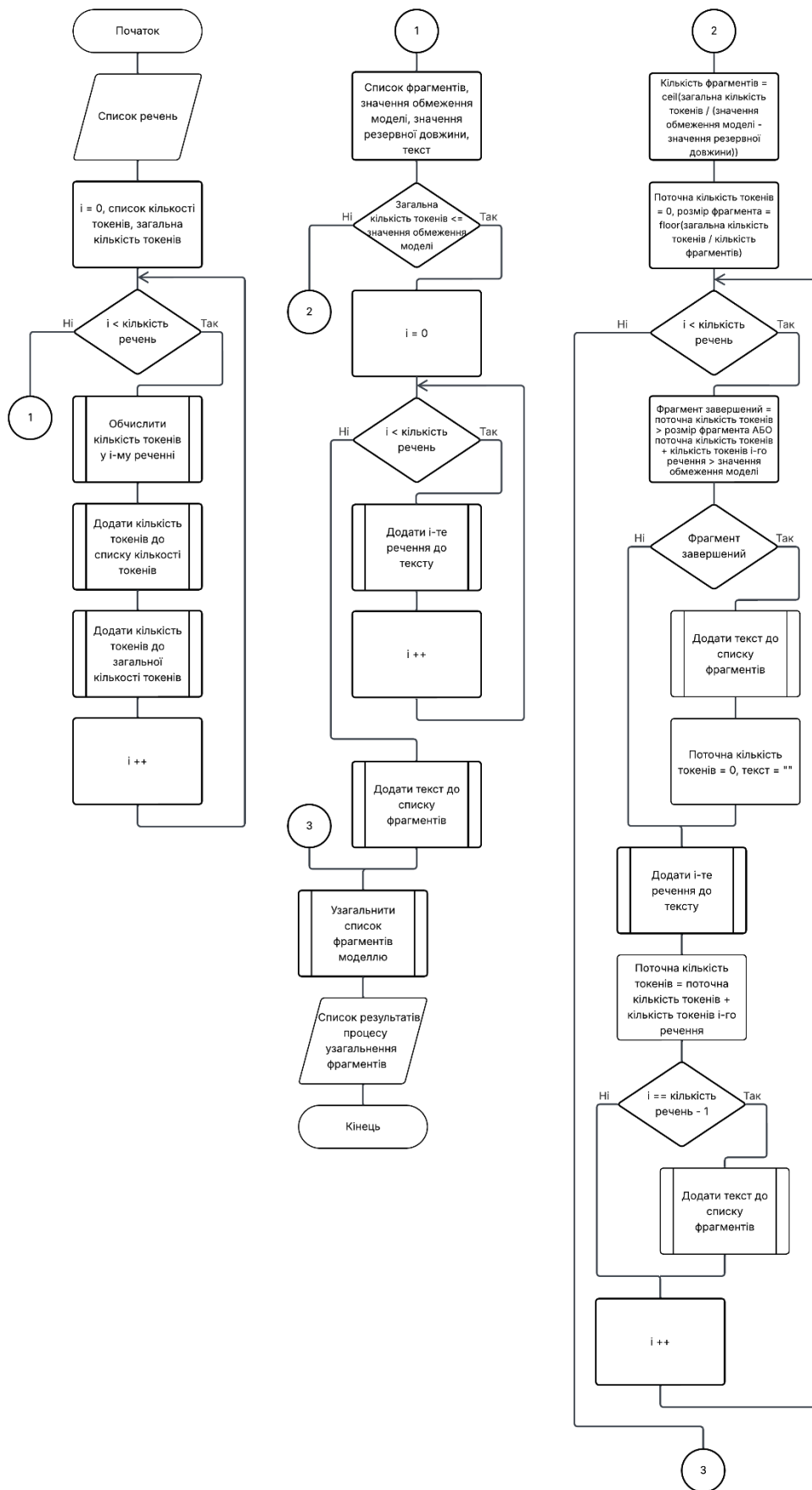


Рис. 2.2. Алгоритм абстрактного узагальнення з застосуванням фрагментації визначеної вмістом

## **2.4. Вибір метрик для визначення кількісних характеристик ефективності пропонованого методу**

Для оцінювання ефективності методів узагальнення використовують як прості кількісні метрики на кшталт швидкості, так і кількісні метрики, що характеризують якість результату узагальнення. Зважаючи на мету розробки комбінованого методу, а саме можливість швидкої обробки тексту на користувацьких пристроях, швидкість роботи є основною метрикою для перевірки методу. Оскільки пропонований метод забезпечує послідовність і зв'язність результату узагальнення, а також охоплення всіх значущих тем вхідного тексту, а, як було зазначено при огляді наявних підходів та методів узагальнення великорозмірного тексту, лише метод багатокрокового абстрактного узагальнення з застосуванням фрагментації визначеної контекстом забезпечує такі ж якісні характеристики результату узагальнення, порівняння швидкості обробки тексту має проводитися саме з цим методом.

Для перевірки швидкості пропонованого методу автором пропонується використати спрощену реалізацію методу багатокрокового абстрактного узагальнення з застосуванням фрагментації визначеної контекстом. Пропонуються наступні спрощення:

- виконання лише одної ітерації замість кількох – оскільки для досягнення стислого узагальнення великорозмірного тексту зазвичай потрібно як мінімум дві ітерації, така зміна спричинить збільшення швидкості обробки та збільшення розміру результату узагальнення за рахунок зменшення кількості викликів до моделі абстрактного узагальнення;
- заміна фрагментації визначеної контекстом на фрагментацію визначену вмістом – забезпечить подальше збільшення швидкості обробки, адже для виконання фрагментації визначеної контекстом необхідне застосування моделі

машинного навчання, що збільшить загальний час виконання методу.

Дані спрощення значно прискорять виконання методу для порівняння, тож якщо при проведенні оцінювання пропонований метод буде мати вищу швидкість обробки тексту, ніж спрощений метод для порівняння, він також буде швидшим за оригінальний метод багатокрокового абстрактного узагальнення з застосуванням фрагментації визначеної контекстом. Очевидно, що спрощення також впливають на результат узагальнення, адже результат буде менш стислим і відповідно міститиме більше деталей, однак для порівняння швидкості роботи методів це не є актуальним.

Оскільки і пропонований метод, і метод для порівняння використовуватимуть однакові моделі абстрактного узагальнення, швидкість узагальнення буде вимірюватися за наступною формулою:

$$V = N/t, \quad (2.1)$$

де:

- $N$  – загальна кількість tokenів вхідного тексту для моделі абстрактного узагальнення;
- $t$  – загальний час виконання методу.

Використання саме tokenів, а не інших одиниць, наприклад слів чи речень, обумовлене тим, що токени є найменшою логічною одиницею в процесі абстрактного узагальнення, а отже при абстрактному узагальненні з застосуванням фрагментації кількість викликів моделі машинного навчання напряду залежить від кількості tokenів у тексті. Оскільки найбільший відсоток часу в процесі узагальнення складає робота моделі машинного навчання, а основне підвищення швидкості пропонованого методу досягається за рахунок зменшення кількості викликів до вищезгаданої моделі, обраний спосіб порівняння швидкості найкраще відобразить різницю між пропонованим та спрощеним еталонним методами.

На рис. 2.3 зображено загальну схему спрощеного методу для порівняння з пропонованим комбінованим методом.



Рис. 2.3. Схема етапів спрощеного методу багатокрокового абстрактного узагальнення

Оцінювання якісних характеристик результату пропонованого методу буде проведене вручну, адже автоматизовані кількісні метрики для представлення якості результату узагальнення мають значні недоліки в контексті узагальнення великорозмірних текстів. Розглянемо найпопулярніші метрики для оцінювання якості результату узагальнення.

1. ROUGE [33] – сімейство метрик, що вимірюють ступінь подібності між автоматично згенерованим узагальненням та одним або кількома еталонними узагальненнями. Включає ROUGE-N, що оцінює збіг послідовностей з  $n$  слів, ROUGE-L, що базується на довжині найдовшої спільної підпослідовності, ROUGE-W, що базується на зважуванні суміжних збігів найдовших спільних підпослідовностей, та ROUGE-S, що оцінює збіг біграм з пропусками. Основна ідея полягає в тому, що чим більше слів та їх послідовностей із згенерованого узагальнення збігаються з еталонним узагальненням, тим вища якість згенерованого узагальнення.

Переваги:

- простота та швидкість оцінювання результату узагальнення;
- є стандартною метрикою у сфері обробки природномовних даних.

Недоліки:

- вимірювання збігів між конкретними словами, зниження оцінки у випадку перефразувань, зміни порядку слів чи використання синонімів;
- чутливість алгоритму до довжини згенерованого узагальнення;
- визначення оцінки як середнього значення від оцінок порівняння згенерованого узагальнення з кількома еталонними узагальненнями;

- немає можливості оцінити зміст згенерованого узагальнення.

2. BLEU [34] – метрика, що була спершу розроблена для оцінювання машинного перекладу, але також застосовується для оцінювання результату узагальнення. BLEU обчислює точність n-грам, порівнюючи частку збігів у згенерованому узагальненні, а також застосовує множинне згладжування для зниження оцінки за надмірну повторюваність та штраф за надмірну стислість для недопуску надто коротких узагальнень.

Переваги:

- простота та швидкість оцінювання результату узагальнення;
- як і ROUGE є стандартною метрикою у сфері обробки природномовних даних.

Недоліки:

- відсутнє врахування синонімів, семантичного значення слів та контексту їх використання;
- зниження оцінки у випадку перефразувань або зміни порядку слів;
- немає можливості оцінити зміст згенерованого узагальнення.

3. METEOR [35] – це метрика, розроблена для поліпшення якості автоматичного оцінювання результатів машинного перекладу, яка також успішно застосовується для оцінювання якості автоматичних узагальнень. На відміну від ROUGE та BLEU, METEOR використовує вирівнювання слів між згенерованим узагальненням та еталонним узагальненням з урахуванням синонімів за допомогою лінгвістичного ресурсу WordNet [36], стемінгу, а також порядку слів, що дає змогу краще відображати змістову подібність. Однак подібне застосування лінгвістичних

ресурсів без врахування контексту все ще не дає можливості надійно оцінити якість згенерованого узагальнення.

Переваги:

- врахування синонімів та порядку слів, застосування стемінгу;
- менша залежність від точних збігів;
- серед оцінок порівняння з кількома еталонними узагальненнями обирається найкраща оцінка.

Недоліки:

- працює лише на лексичному рівні, а не на рівні значення контексту;
- може занижувати оцінку, якщо формулювання дуже відрізняється, хоч і зберігає суть;
- має нижчу швидкість ніж простіші метрики на кшталт ROUGE.

4. BERTScore [37] – це метрика, яка використовує контекстуалізовані векторні відображення слів, отримані з моделі-трансформера BERT, для обчислення семантичної подібності між згенерованим та еталонним узагальненням, завдяки чому вона досягає високої кореляції з людськими оцінками. Основна ідея метрики полягає в тому, що замість підрахунку точних збігів слів чи їх послідовностей, BERTScore обчислює векторне відображення для кожного слова в обох узагальненнях, після чого порівнює вектори слів за допомогою косинусної подібності.

Переваги:

- врахування контексту слів за допомогою векторних відображень обчислених моделлю-трансформером;
- менша чутливість до перефразувань та перестановок;

- можливість тренування моделі для використання на текстах вузької спеціалізації.

Недоліки:

- немає явного врахування порядку слів у реченні;
- менша швидкість роботи та більша ресурсозатратність через використання моделі.

Як бачимо, метрики на кшталт ROUGE та BLEU були створені досить давно, тому є широкоживаними, але з тої ж причини вони вже втратили свою актуальність, адже мають надто багато недоліків для оцінювання сучасних засобів узагальнення. Конкретні проблеми ROUGE, більшість яких також стосуються і BLEU, розглядаються у статті щодо обмежень автоматичного узагальнення з точки зору ROUGE [38]. METEOR, хоч і враховує синонімію, також не в змозі правильно оцінити результат узагальнення, адже не враховує контекст слів. З точки зору врахування контексту BERTScore є найкращим варіантом з чотирьох, однак було визначено, що навіть найновіші моделі не можуть досягнути рівня людини при оцінці результатів абстрактного узагальнення [39]. Також є очевидним те, що дані метрики не здатні оцінити зв'язність та послідовність результату узагальнення, а лише певною мірою оцінюють тематичне охоплення вхідного тексту.

Окрім того, для великорозмірних текстів наявна істотна проблема формулювальної неоднорідності та варіативності еталонних узагальнень, адже через кількість інформації у тексті та кількість способів її перефразування навіть узагальнення створені людьми можуть суттєво відрізнитися один від одного, хоч і відображають одні й ті ж тематичні блоки. Те ж стосується і автоматизованих методів узагальнення великорозмірних текстів. Для проведення автоматизованого оцінювання для кожного великорозмірного тексту потрібно було б мати велику кількість еталонних узагальнень, що різнопланово висвітлюють інформацію у ньому, щоб обрати найкращу оцінку з кількох, як це робиться в METEOR, і не

занизити її для правильно згенерованого узагальнення. Однак на даний момент не існує набору даних з великорозмірними науково-популярними текстами та багатьма різноплановими узагальненнями, що добре відображають суть тексту та його основні теми, як і не існує достатньої кількості перевірених методів узагальнення великорозмірного тексту, щоб згенерувати різнопланові еталонні узагальнення. Через це об'єктивне оцінювання якості результату узагальнення великорозмірних науково-популярних текстів за допомогою автоматизованих метрик є неможливим.

Зважаючи на вищесказане та на те, що для обчислення векторних відображень та абстрактного узагальнення запропонований метод використовує моделі машинного навчання, що є одними з останніх розробок у сфері обробки природномовних даних та мають одні з найкращих результатів за кількісними метриками щодо якості результату їх роботи, якість результату має бути співставною з якістю результату обчислення методом багатокрокового абстрактного узагальнення з застосуванням фрагментації визначеної контекстом, а оцінювання якості результату узагальнення, а саме повноти тематичного охоплення, зв'язності та послідовності, буде проведена вручну через неможливість об'єктивної автоматизованої перевірки за допомогою наявних метрик.

## **2.5. Висновки до розділу**

У даному розділі було розглянуто основу першого кроку запропонованого методу, а саме застосування методу екстрактного узагальнення з використанням кластеризації. На основі розглянутих недоліків даного методу було запропоновано наступні способи покращення:

- використання методу обчислення векторних відображень, що враховує контекст слів та їх порядок в реченні;
- використання методу кластеризації, що не залежить від початкового задання кількості кластерів та здатний обробляти структури даних різної форми та розміру;

- обрання всіх речень з кластера, що мають ймовірність приналежності до кластера більшу або рівну 0.8, для запобігання неповного або хибного представлення тематичного блоку, та у послідовності, визначеній вхідним текстом, для забезпечення якісної характеристики результату узагальнення “послідовність”.

Також було розглянуто основу другого кроку пропнованого методу, а саме застосування методу абстрактного узагальнення з поділом тексту на фрагменти. Було обґрунтовано вибір фрагментації визначеної вмістом на основі результатів попереднього кроку.

Окрім того було описано пропонований комбінований метод узагальнення великорозмірних науково-популярних текстів, що складається з наступних кроків:

1. Попередня обробка тексту.
2. Токенізація тексту.
3. Обчислення векторних відображень.
4. Кластеризація векторних відображень.
5. Обрання речень для абстрактного узагальнення.
6. Застосування моделі абстрактного узагальнення.
7. Сортування результатів.
8. Конкатенація результатів.

Наостанок було обрано метрику для оцінювання ефективності методу, а саме швидкість роботи, що вимірюється як загальна кількість токенів у вхідному тексті поділена на загальний час роботи методу, запропоновано спрощений метод для порівняння швидкості роботи та обґрунтовано недоцільність автоматизованого оцінювання якості результату узагальнення через проблеми, зв’язані з розмірністю тексту.

### **3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ КОМБІНОВАНОГО МЕТОДУ УЗАГАЛЬНЕННЯ ВЕЛИКОРОЗМІРНИХ НАУКОВО-ПОПУЛЯРНИХ ТЕКСТІВ**

#### **3.1. Обґрунтування вибору засобів реалізації програмного забезпечення**

Для демонстрації роботи запропонованого комбінованого методу узагальнення великорозмірних науково-популярних текстів було створено програмну реалізацію у вигляді консольного додатку із можливістю запуску з параметрами командного рядка. Для розробки використовувалися такі технології:

- мова програмування Python;
- лінгвістичний програмний пакет NLTK для розбиття тексту на речення;
- програмну бібліотеку TensorFlow для роботи з датасетами та програмну бібліотеку TensorFlowHub для завантаження моделей машинного навчання;
- програмна бібліотека для кластеризації HDBSCAN;
- програмну бібліотеку HuggingFace Transformers для використання моделей машинного навчання.

##### ***3.1.1. Тип реалізації програмного забезпечення***

Рішення про реалізацію програмного забезпечення у вигляді консольного додатку з інтерфейсом командного рядка було прийнято з урахуванням специфіки задачі та вимог до гнучкості та інтеграційної здатності системи.

Узагальнення великорозмірних текстів – задача, що входить до ширшого класу завдань обробки природномовних даних і зазвичай виступає як окремий етап у рамках більших інформаційно-аналітичних систем, таких як платформи для моніторингу ЗМІ, автоматизовані системи аналізу

документів або інтелектуальні помічники. Внаслідок цього розроблюване програмне забезпечення має мати можливість інтеграції в такі системи і не потребує власного графічного інтерфейсу, що найчастіше досягається за допомогою створення консольного додатку. Такі додатки зазвичай надають можливість налаштування, наприклад вибір моделі абстрактного узагальнення для забезпечення необхідних користувачу характеристик результату узагальнення, тоді як процес її завантаження та конфігурації відбувається автоматично. Консольний додаток, який приймає параметри у вигляді шляху до вхідного файлу та назви моделі абстрактного узагальнення, дозволяє швидко налаштовувати систему, проводити експерименти та тести і адаптувати її до різних сценаріїв використання. Розроблення консольного додатку надає низку переваг для розробки, тестування, інтеграції та швидкого прототипування:

- консольна форма додатку мінімізує накладні витрати на створення інтерфейсу користувача, що дозволяє зосередитись на реалізації алгоритмів узагальнення;
- консольний додаток легко автоматизується та інтегрується у скрипти й пайплайни обробки даних інформаційно-аналітичних систем;
- консольна архітектура спрощує процес відлагодження та тестування, що особливо важливо при порівнянні результатів, отриманих за допомогою різних типів моделей машинного навчання.

Отже, реалізація методу узагальнення великорозмірних науково-популярних текстів у вигляді консольного додатку є обґрунтованим вибором як з точки зору ефективності розроблення, так і з погляду практичного застосування в складі більш складних систем аналізу природномовних даних.

### ***3.1.2. Мова програмування Python***

Python є високорівневою мовою програмування загального призначення, основною метою якої є забезпечення пришвидшення вивчення, легкості написання та високої читабельності коду. Вона підтримує декілька парадигм програмування, зокрема структурну, об'єктно-орієнтовану та функціональну, і характеризується динамічною типізацією, що дозволяє зменшити обсяг шаблонного коду. Завдяки наявності розвиненої стандартної бібліотеки та численних зовнішніх пакетів та бібліотек, на сьогодні Python є однією з найпоширеніших мов програмування, яка є найпопулярнішою в таких галузях, як обробка великих даних, розробка систем та моделей машинного навчання, Інтернет речей, тощо. Python має низку властивостей, які визначають його як зручний інструмент для широкого кола завдань:

- Мова загального призначення – Python не спеціалізується на якійсь одній предметній області, що дозволяє ефективно використовувати його в різноманітних технологічних сферах від веброзробки до побудови прототипів і реалізації систем машинного навчання.
- Інтерпретованість – програми, написані на Python, виконуються у віртуальному середовищі Python Virtual Machine, що інтерпретує байткод у режимі реального часу. Це забезпечує зручність розроблення, кросплатформеність і автоматичне керування пам'яттю, хоча й може призводити до зниження продуктивності порівняно з компільованими мовами.
- Об'єктно-орієнтованість – Python підтримує об'єктно-орієнтоване програмування, яке дозволяє моделювати прикладні домени у вигляді класів і об'єктів та краще структурувати код. Разом з тим, мова також надає засоби функціонального програмування, що розширює можливості проєктування.

- Високий рівень абстракції – Python належить до високорівневих мов, а отже дозволяє розробнику оперувати на рівні логіки задачі, а не низькорівневих операцій із пам'яттю або апаратною частиною.
- Динамічна типізація – типи змінних у Python визначаються під час виконання програми, що спрощує початкову розробку та розширення функціональності програмного забезпечення, однак може створювати проблеми при масштабуванні великих проєктів без належної перевірки типів.
- Портативність – завдяки моделі “Write Once, Run Anywhere” код на Python можна запускати без змін на більшості сучасних платформ, що спрощує розгортання та переносимість програмного забезпечення.
- Простота у засвоєнні – лаконічний та інтуїтивно зрозумілий синтаксис робить Python зручним для початківців, адже серед найпопулярніших мов програмування він найбільше схожий на звичайні команди англійською мовою.
- Розвинена екосистема – наявність широкого спектру пакетів, бібліотек і фреймворків забезпечує швидку та спрощену розробку для будь-якої предметної області.

При виборі мови програмування значний вплив на рішення мала наявність таких пакетів та бібліотек, як NumPy, Pandas та TensorFlow, а також спеціалізованих засобів для оброблення природної мови, таких як NLTK, що значно розширюють можливості мови щодо обробки даних. Оскільки обробка природномовних даних є складним завданням, що вимагає застосування моделей машинного навчання, у процесі реалізації програмного забезпечення для запропонованого методу узагальнення великорозмірного науково-популярного тексту особливу роль відіграла доступність бібліотек на кшталт HuggingFace Transformers, що забезпечує доступ до найновіших моделей. З огляду на доступність інструментів,

простоту розробки та продуктивність, Python було обрано як мову програмування для реалізації програмного забезпечення в межах даної роботи.

### ***3.1.3. Лінгвістичний програмний пакет NLTK***

Natural Language Toolkit (NLTK) – це потужний і широко використовуваний інструментарій для оброблення природної мови, реалізований мовою програмування Python [40]. Він розроблений з акцентом на простоту інтеграції в освітні, дослідницькі та прототипувальні проекти, що охоплюють широкий спектр задач у сфері комп'ютерної лінгвістики. Пакет забезпечує доступ до понад 50 корпусів і лексичних ресурсів, включаючи WordNet, а також містить велику кількість алгоритмів і функцій для розв'язання ключових задач оброблення тексту: класифікації, токенизації, стемінгу, лематизації, тегування частин мови, синтаксичного аналізу, семантичної інтерпретації тощо. Одним з ключових факторів популярності NLTK є його модульна структура та підтримка стандартних підходів до мовної обробки, що дозволяє легко поєднувати різні компоненти для досягнення конкретних цілей дослідження.

У межах даної роботи було застосовано функціональний модуль NLTK English Pickle Punkt Sentence Tokenizer. Одним із базових етапів аналізу природномовного тексту є поділ його на окремі речення – процедура, що передує багатьом іншим операціям, таким як аналіз граматичної структури або генерація векторних відображень. Для цієї мети в NLTK реалізовано механізм Punkt Sentence Tokenizer – статистичний токенізатор, який виявляє межі речень у тексті на основі знаків пунктуації. У NLTK доступна спеціалізована модель для англійської мови у вигляді серіалізованого об'єкта, що зберігає структуровані правила та ймовірнісні шаблони пунктуації. Цей токенізатор здатен обробляти складні випадки, зокрема, правильне розділення речень із аббревіатурами, датами, числовими виразами або прямою мовою. Він є критично важливим при роботі з

довгими документами або корпусами текстів, де надійність поділу на речення визначає точність подальших лінгвістичних процедур.

NLTK вважається одним із найбільш доступних і функціонально повних інструментів для лінгвістичного аналізу текстів. Його використання виправдане у випадках, коли необхідно швидко реалізувати і протестувати алгоритми обробки природномовних даних, а також мати контроль над процесом обробки на всіх його етапах. Велика кількість супровідної документації, наявність вбудованих корпусів і відкритість платформи сприяли вибору саме NLTK для реалізації програмного забезпечення.

### ***3.1.4. Програмна бібліотека TensorFlow та платформа TensorFlow Hub***

TensorFlow – це потужна платформа з відкритим вихідним кодом для створення та навчання моделей машинного навчання, що була розроблена дослідницьким підрозділом Google Brain [41]. Бібліотека підтримує як розробку нейронних мереж, так і виконання навчання на різних рівнях абстракції: від низькорівневих тензорних операцій до побудови комплексних багаторівневих архітектур із використанням модульної системи. Однією з ключових переваг TensorFlow є його інтеграція з численними інструментами для обробки даних, зокрема з об'єктами типу DataFrame через бібліотеки типу pandas або tensorflow.data, що дозволяє зручно організувати попередню обробку текстової або числової інформації. У межах даної роботи TensorFlow використовувався переважно для обробки вхідних наборів даних, представлених у вигляді структурованих таблиць типу DataFrame. Такий формат дозволив ефективно обчислити векторні відображення для речень.

Для безпосереднього отримання векторних відображень речень було використано платформу TensorFlow Hub – спеціалізоване розширення до базової бібліотеки TensorFlow, яке забезпечує доступ до попередньо навчених моделей машинного навчання [42]. Завдяки єдиному

уніфікованому інтерфейсу, TensorFlow Hub дозволяє завантажувати і застосовувати складні моделі без необхідності повторного навчання чи налаштування низькорівневих параметрів. Однією з моделей, інтегрованих у TensorFlow Hub, є Universal Sentence Encoder (USE) – модель для кодування речень у вектори фіксованої розмірності. На відміну від класичних підходів, які працюють на рівні окремих слів, USE дозволяє отримати семантично інформативне векторне відображення речення, зберігаючи як синтаксичні, так і контекстні залежності. Завантаження моделі відбувалося безпосередньо з репозиторію TensorFlow Hub, що спростило процес інтеграції та дало змогу уникнути витрат на повторне тренування.

Використання TensorFlow у зв'язці з TensorFlow Hub забезпечило ряд переваг у реалізації методу:

- ефективне управління датафреймом векторних відображень речень;
- зручне використання попередньо натренованих моделей через стандартизований інтерфейс;
- сумісність із іншими популярними бібліотеками Python, зокрема numpy, pandas та sklearn.

Таким чином, поєднання можливостей TensorFlow як системи керування даними та TensorFlow Hub як платформи доступу до попередньо натренованих моделей, зокрема Universal Sentence Encoder, дозволило ефективно реалізувати задачу обчислення та управління векторними відображеннями речень.

### ***3.1.5. Програмна бібліотека для кластеризації HDBSCAN***

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) – це спеціалізована програмна бібліотека для ієрархічної щільнісної кластеризації, яка реалізує однойменний алгоритм кластерного аналізу [28]. Пакет створений як вдосконалення класичного

алгоритму кластеризації DBSCAN з метою підвищення адаптивності до складних структур даних, неоднорідної щільності та точок-викидів. Оригінальна бібліотека HDBSCAN містить повноцінну, оптимізовану реалізацію алгоритму з широкими можливостями налаштування, підтримкою ієрархічного кластерного дерева та покращеною продуктивністю завдяки використанню Cython. Бібліотека HDBSCAN надає розширений інструментарій для виконання кластеризації даних на основі ієрархічної щільності з такими ключовими особливостями:

- автоматичне визначення кількості кластерів – на відміну від багатьох традиційних алгоритмів, таких як k-means, HDBSCAN не вимагає заздалегідь задавати кількість кластерів, а формує кластери динамічно на основі щільності розподілу даних, що робить його ідеальним для задач, де кількість кластерів заздалегідь невідома;
- підтримка ієрархічної структури кластерів – алгоритм будує дерево кластерної ієрархії, з якого автоматично відбирається оптимальне плоске розбиття даних, виходячи з критерію стабільності кластерів;
- виявлення точок-викидів – однією з важливих властивостей бібліотеки є здатність класифікувати частину даних як викиди, або ж шум, тобто такі, що не належать до жодного щільного кластера;
- оцінювання достовірності кластеризації – для кожної точки бібліотека обчислює ймовірнісну міру приналежності до відповідного кластера, що дозволяє оцінювати достовірність кластеризації і застосовувати додаткову фільтрацію даних;
- підтримка кастомних метрик – HDBSCAN дозволяє використовувати як стандартні метрики відстані, як евклідова або косинусна, так і метрики визначені користувачем, що дає змогу адаптувати алгоритм до специфіки даних;

- оптимізована продуктивність – реалізація побудована на основі Cython, що забезпечує значне прискорення обчислень, особливо при роботі з великими обсягами даних або даними з великою вимірністю.

У межах даної роботи бібліотека HDBSCAN використовувалася для кластеризації векторних відображень речень для виявлення тематично подібних груп речень. Такий підхід дозволив структурувати текстовий масив, ідентифікувати основні тематичні блоки, а також виключити із аналізу речення, які не належали до жодної смислової групи. Завдяки ефективній кластеризації високовимірних даних, надійному виявленню шуму, підтримці адаптивної щільності, високій гнучкості при налаштуванні та розширеному функціоналу кластерного аналізу HDBSCAN зарекомендував себе як потужний і зручний інструмент для кластеризації.

### ***3.1.6. Програмна бібліотека HuggingFace Transformers***

HuggingFace Transformers – це відкрита програмна бібліотека для роботи з трансформерними моделями, яка надає зручний інтерфейс для використання попередньо натренованих мовних моделей з різноманітними архітектурами, такими як BERT, GPT, T5, RoBERTa, BART та PEGASUS [43]. Розроблена компанією Hugging Face, ця бібліотека швидко стала галузевим стандартом у сфері оброблення природномовних даних, машинного та глибокого навчання завдяки поєднанню потужності, гнучкості та зручності у використанні.

Бібліотека transformers реалізує високорівневий API для завантаження, налаштування та застосування трансформерних моделей, а також інтегрується з провідними фреймворками машинного навчання, такими як PyTorch і TensorFlow. Вона включає у себе тисячі попередньо навчених моделей, доступних через Hugging Face Hub – централізоване сховище моделей із метаданими, документацією та прикладами використання. Однією з найважливіших можливостей бібліотеки є

підтримка роботи з попередньо натренованими трансформерами, які можуть бути використані “як є” або адаптовані до власних задач за допомогою донавчання. Крім того, бібліотека забезпечує наступну функціональність:

- автоматичне завантаження та кешування моделей;
- оптимізовану токенизацію через класи `Tokenizer` і `PreTrainedTokenizer`;
- прозоре керування ресурсами;
- можливість генерації, узагальнення, перекладу тексту з використанням моделей на основі архітектур `encoder-decoder`.

У рамках даної роботи бібліотека `HuggingFace Transformers` була використана для отримання та використання попередньо навчених мовних моделей `BART` та `PEGASUS`, які застосовувалися для автоматичного узагальнення текстів. Обидві ці моделі реалізують архітектуру трансформера типу “`encoder-decoder`” і спеціалізуються на задачах генеративного характеру, таких як скорочення та переформулювання змісту. `BART` – модель, що поєднує переваги двонаправленого кодування та авто-регресивного декодування. Завдяки цьому, вона добре підходить для задач, де необхідно трансформувати текст з урахуванням повного контексту. `PEGASUS` – спеціалізована модель для узагальнення, яка використовує унікальний підхід до навчання з пропущеними фрагментами і демонструє високі результати на різних тестувальних наборах. Завдяки використанню `transformers`, моделі `BART` і `PEGASUS` можна було завантажити, ініціалізувати та застосовувати для узагальнення текстів із мінімальними зусиллями, зберігаючи при цьому повну гнучкість у налаштуванні параметрів генерації. Бібліотека має наступні переваги:

- простота використання – завдяки уніфікованому API, навіть складні моделі можна використовувати за допомогою кількох рядків коду;

- потужність і продуктивність – моделі легко масштабуються та можуть обробляти великі об’єми тексту з високою швидкістю;
- актуальність і підтримка спільноти – HuggingFace Transformers активно розвивається, має повноцінну документацію та велику спільноту користувачів, що робить її зручним вибором для дослідників;
- гнучкість – бібліотека підтримує як звичайне використання моделей, так і повний цикл навчання моделей на власних даних.

Таким чином, використання бібліотеки HuggingFace Transformers дозволило ефективно інтегрувати сучасні трансформерні моделі абстрактного узагальнення в аналітичний процес, спростити роботу з ними та забезпечити високу якість результатів без потреби навчання моделей з нуля.

### 3.2. Архітектура розробленого програмного забезпечення

Програмне забезпечення, що реалізує пропонуваній комбінований метод узагальнення великорозмірних науково-популярних текстів, написано на мові програмування Python у формі консольного додатку з підтримкою параметрів командного рядка. Модульна архітектура розробленого програмного забезпечення зображена на рис. 3.1.

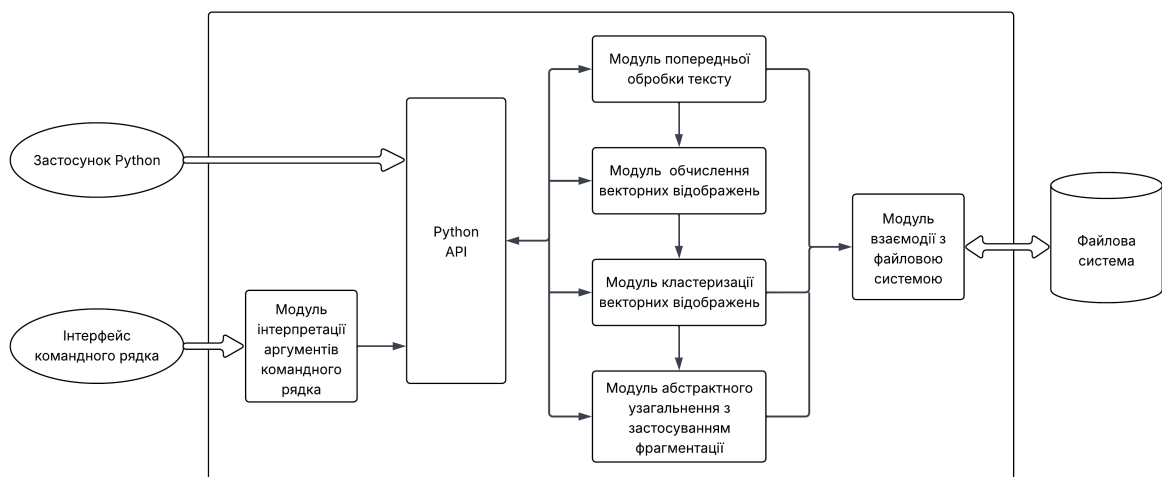


Рис. 3.1. Діаграма модулів розробленого програмного забезпечення

В програмній реалізації можна виділити наступні модулі:

1. *Python API*. Уможлиблює застосування вбудованих модулів Python, виконання базових операцій з даними, підключення додаткових пакетів і бібліотек.
2. *Модуль інтерпретації аргументів командного рядка*. Дозволяє запускати розроблений додаток за допомогою командного рядка та вказувати аргументи попередньо визначених параметрів, що включають шлях до файлу для узагальнення та назву моделі абстрактного узагальнення, що буде застосована. Після інтерпретації аргументів ініціалізує відповідні змінні та запускає основний код програми.
3. *Модуль попередньої обробки тексту*. Відповідає за вилучення шумових символів, зайвих пробілів, табуляцій та переходів на новий рядок, тобто приведення тексту до форми одного рядка. Окрім того в межах попередньої обробки проводиться токенизація (розбиття) тексту на речення за допомогою бібліотеки NLTK та вилучення всіх речень, що містять менше ніж 30 символів.
4. *Модуль обчислення векторних відображень*. Відповідає за генерацію векторних відображень для кожного речення з отриманого списку. Модуль використовує платформу TensorFlow Hub для завантаження моделі USE, що обчислює векторні відображення, та повертає список векторів у формі об'єкта TensorFlow DataFrame.
5. *Модуль кластеризації векторних відображень*. Відповідає за кластеризацію векторних відображень, отриманих з модуля обчислення векторних відображень. Використовує бібліотеку `hdbscan` для виконання кластеризації. Після завершення кластеризації речення, що належать одному кластеру, конкатенуються, а список з отриманих уривків, що

відображають теми вхідного тексту, передається в наступний модуль.

6. *Модуль абстрактного узагальнення з застосуванням фрагментації.* Відповідає за проведення фрагментації отриманих уривків та узагальнення їх фрагментів моделлю абстрактного узагальнення, обраною користувачем при запуску додатку. Доступні моделі завантажуються та конфігуруються за допомогою бібліотеки HuggingFace Transformers. Результати узагальнення фрагментів уривків конкатенуються в кінцевий результат.
7. *Модуль взаємодії з файловою системою.* Відповідає за зчитування файлів із вхідними текстовими даними та запис результатів роботи алгоритму та логування у вигляді текстових файлів та їх збереження у файловій системі. Зчитування та запис відбувається у форматі UTF-8 для забезпечення коректного відображення при наявності у вхідному тексті символів, що не належать до кодування ASCII.

### **3.3. Особливості реалізації програмного забезпечення**

Для забезпечення коректної роботи програмного забезпечення, що включає відсутність втрати частини даних або падінь з помилкою, а також правильний поділ тексту на речення, правильну послідовність з'єднання узагальнених кластерних уривків, тощо, під час реалізації запропонованого методу були прийняті рішення, описані в наступних підрозділах.

#### ***3.3.1. Використання бібліотеки NLTK для розбиття тексту на речення***

Для звичайного тексту розмовною мовою зазвичай достатньо проводити розбиття на речення орієнтуючись лише на знаки пунктуації, що завершують речення, тобто на крапку, знак оклику та знак питання. Однак

у науково-популярних текстах, хоч і в меншій мірі, ніж в нехудожній літературі, зустрічаються скорочення, що містять крапки, та потребують спеціальної обробки. З цією метою при використанні засобу English Pickle Punkt Tokenizer з бібліотеки NLTK в множину відомих токенизатору аббревіатур були додані стандартні аббревіатури “i.e.” та “e.g.” (лістинг 3.1), що часто зустрічалися у тестових вхідних текстах. Приклади токенизації тексту без модифікації (лістинг 3.2) та з модифікацією (лістинг 3.3) наведені нижче. Як бачимо, без повноцінного врахування широковживаних аббревіатур розподіл тексту на речення буде виконаний неправильно, що й вимагає модифікації параметрів токенизатора.

### Лістинг 3.1. Модифікація стандартного токенизатора з додаванням нових аббревіатур

```
# split text into sentences
sentence_tokenizer =
nltk.data.load('tokenizers/punkt/english.pickle')
sentence_tokenizer._params.abbrev_types =
    set.union(sentence_tokenizer._params.abbrev_types,
              {'i.e.', 'e.g.'})
all_sentences = sentence_tokenizer.tokenize(clean_text)
```

### Лістинг 3.2. Приклад розділу тексту на речення без модифікації параметрів токенизатора

```
"Many programming languages support object-oriented concepts, e.g.
Java, C++ and Python." ->
[
  "Many programming languages support object-oriented concepts, e.",
  "g.",
  "Java, C++ and Python."
]
```

### Лістинг 3.3. Приклад розділу тексту на речення з модифікацією параметрів токенизатора

```
"Many programming languages support object-oriented concepts, e.g.
Java, C++ and Python." ->
[
  "Many programming languages support object-oriented concepts,
  e.g. Java, C++ and Python."
]
```

### 3.3.2. Використання моделі USE для обчислення векторних відображень речень

Для обчислення векторних відображень речень модель USE приймає список речень та одразу завантажує весь список у пам'ять. Для пропонованого методу це не є прийнятним, адже одна з вимог – це можливість виконання на користувацьких пристроях, а завантаження у пам'ять великого списку (приблизно 7000 речень) на середньостатистичному пристрої призводить до передчасного завершення роботи коду з помилкою. Для запобігання подібної ситуації було прийнято рішення передавати речення в модель партіями по 1000 штук (лістинг 3.4). На списку розміром 6000 речень було експериментально перевірено, що різниці у часі між одночасною та почерговою обробкою речень немає, тож даний спосіб не завдає шкоди загальній швидкості виконання методу.

Лістинг 3.4. Обчислення векторних відображень партіями для запобігання перевантаження пам'яті

```
# generate vector representations (embeddings) for sentences

embedding_model = hub.load('../models/use-large-tf2')
embeddings = None

for i in range(
    math.ceil(len(sentences)/sentence_embedding_block_max_size)
):
    temp_embeddings = embedding_model(
        sentences[
            (i * sentence_embedding_block_max_size):
            ((i + 1) * sentence_embedding_block_max_size)
        ]
    )
    if (i == 0):
        embeddings = temp_embeddings
    else:
        embeddings = tf.concat(
            [embeddings, temp_embeddings],
            axis = 0
        )
```

### **3.3.3. Використання бібліотеки HDBSCAN для кластеризації векторних відображень**

Отримавши векторні відображення від моделі USE, переходимо до кластеризації, щоб згрупувати векторні відображення, а отже і речення, за їхньою тематикою. Для кластеризації необхідна можливість порівняння двох векторів, зазвичай у сфері обробки природномовних даних це досягається шляхом обчислення косинусної подібності, що стає в пригоді, якщо вектори не нормалізовані, а саме по собі значення відстані не несе смислового навантаження. Однак модель USE повертає нормалізовані вектори, а для задачі кластеризації порівняння відстаней між точками не має недоліків порівняно з мірою косинусної подібності, для якої необхідні додаткові обчислення, тож на вхід методу кластеризації передається список векторів без будь-яких додаткових обчислень (лістинг 3.5). Завдяки використанню алгоритму HDBSCAN, що правильно обробляє точки-викиди, у кластери потрапляє в середньому 10% зі всіх речень наявних у вхідному тексті, що забезпечує значне зменшення кількості викликів моделі абстрактного узагальнення, а отже й зменшення загального часу та збільшення швидкості виконання, на чому й базується покращена ефективність пропонованого методу.

#### **Лістинг 3.5. Виклик методу кластеризації HDBSCAN**

```
# cluster the embeddings

clusterer = hdbscan.HDBSCAN(
    min_cluster_size = 10,
    min_samples = 2,
    cluster_selection_method = 'leaf',
    allow_single_cluster=True
).fit(embeddings)
```

### **3.3.4. Використання моделей абстрактного узагальнення**

Для застосування абстрактного узагальнення було обрано три реалізації моделей машинного навчання, що показали найкращі результати під час їх експериментального оцінювання:

- *pegasus-large* – реалізація моделі PEGASUS, що має обмеження на вхідну послідовність розміром в 1024 токени та тренована на датасетах C4 та HugeNews [44];
- *bart-large-cnn* – реалізація моделі BART, що має обмеження на вхідну послідовність розміром в 1024 токени та тренована на датасеті CNN Daily Mail [45];
- *bart-large-xsum* – реалізація моделі BART, що має обмеження на вхідну послідовність розміром в 1024 токени та тренована на датасеті XSUM для забезпечення максимально короткого результату узагальнення [46].

Використання різних моделей абстрактного узагальнення забезпечує різні характеристики результату узагальнення для забезпечення потреб користувача шляхом використання різної архітектури моделей, різних методик тренування та різних наборів даних для тренування, наприклад більшу стислість у випадку використання *bart-large-xsum*, а також різну швидкість роботи моделей, що вплине на загальний час роботи програмного забезпечення та дозволить більш об'єктивно оцінити ефективність методу. Для можливості швидкого додавання нових моделей було розроблено інтерфейс (лістинг 3.6), який реалізують всі класи моделей машинного навчання. Приклад реалізації інтерфейсу для моделі *pegasus-large* наведений нижче (лістинг 3.7).

Лістинг 3.6. Інтерфейс взаємодії з моделлю абстрактного узагальнення

```
import abc
from typing import List

class LLMInterface(metaclass=abc.ABCMeta):

    @classmethod
    def __subclasshook__(cls, subclass):
        return (hasattr(subclass, 'get_input_limit') and
                callable(subclass.get_input_limit) and
                hasattr(subclass, 'get_chunking_cushion') and
                callable(subclass.get_chunking_cushion) and
                hasattr(subclass, 'tokenize')) and
```

### Продовження лістингу 3.6

```
        callable(subclass.tokenize) and
        hasattr(subclass, 'summarize') and
        callable(subclass.summarize) or
        NotImplemented)

@abc.abstractmethod
def get_input_limit(self) -> int:
    raise NotImplementedError

@abc.abstractmethod
def get_chunking_cushion(self) -> int:
    raise NotImplementedError

@abc.abstractmethod
def tokenize(self, text: str) -> List[str]:
    raise NotImplementedError

@abc.abstractmethod
def summarize(self, chunks: List[str]) -> List[str]:
    raise NotImplementedError
```

### Лістинг 3.7. Реалізація інтерфейсу для моделі *pegasus-large*

```
from .llm_interface import LLMInterface
from typing import List
from transformers import
PegasusForConditionalGeneration, PegasusTokenizer

class Pegasus(LLMInterface):

    def __init__(self):
        self.__name: str = 'google/pegasus-large'
        self.__device: str = 'cpu'
        self.__input_limit: int = 1024
        self.__chunking_cushion: int = 150
        self.__tokenizer =
            PegasusTokenizer.from_pretrained(self.__name)

        self.__model = PegasusForConditionalGeneration
            .from_pretrained(self.__name)
            .to(self.__device)

    def get_input_limit(self) -> int:
        return self.__input_limit

    def get_chunking_cushion(self) -> int:
        return self.__chunking_cushion

    def tokenize(self, text: str) -> List[str]:
        return self.__tokenizer.tokenize(text)

    def summarize(self, chunks: List[str]) -> List[str]:
        batch = self.__tokenizer(
            chunks,
            truncation=True,
```

## Продовження лістингу 3.7

```
        padding="longest",
        return_tensors="pt"
    ).to(self.__device)
    result = self.__model.generate(**batch)
    return self.__tokenizer.batch_decode(
        result,
        skip_special_tokens=True
    )
```

### 3.4. Приклад використання розробленого програмного забезпечення

Розроблене програмне забезпечення для узагальнення великорозмірних науково-популярних текстів передбачає взаємодію з користувачем за допомогою інтерфейсу командного рядка. Для взаємодії з додатком за допомогою інтерфейсу командного рядка необхідно, щоб в системі користувача було встановлено Python версії 3.10. Для запуску додатку необхідно викликати виконуваний файл `app.py` і вказати обов'язкові аргументи командного рядка, а за необхідністю – і додаткові.

Користувач може:

- переглянути доступні параметри командного рядка;
- переглянути файл з результатом узагальнення;
- переглянути файл з логами процесу узагальнення;
- запустити процес узагальнення;
- вказати назву моделі абстрактного узагальнення;
- вказати шлях до файлу з вхідним текстом;
- за необхідності вказати шлях до файлу для запису результату узагальнення;
- за необхідності вказати шлях до файлу для запису логів.

Сценарії використання додатку за допомогою інтерфейсу командного рядка зображені на UML діаграмі (рис. 3.2), окрім того наведено приклад запуску розробленого програмного забезпечення (рис. 3.3).

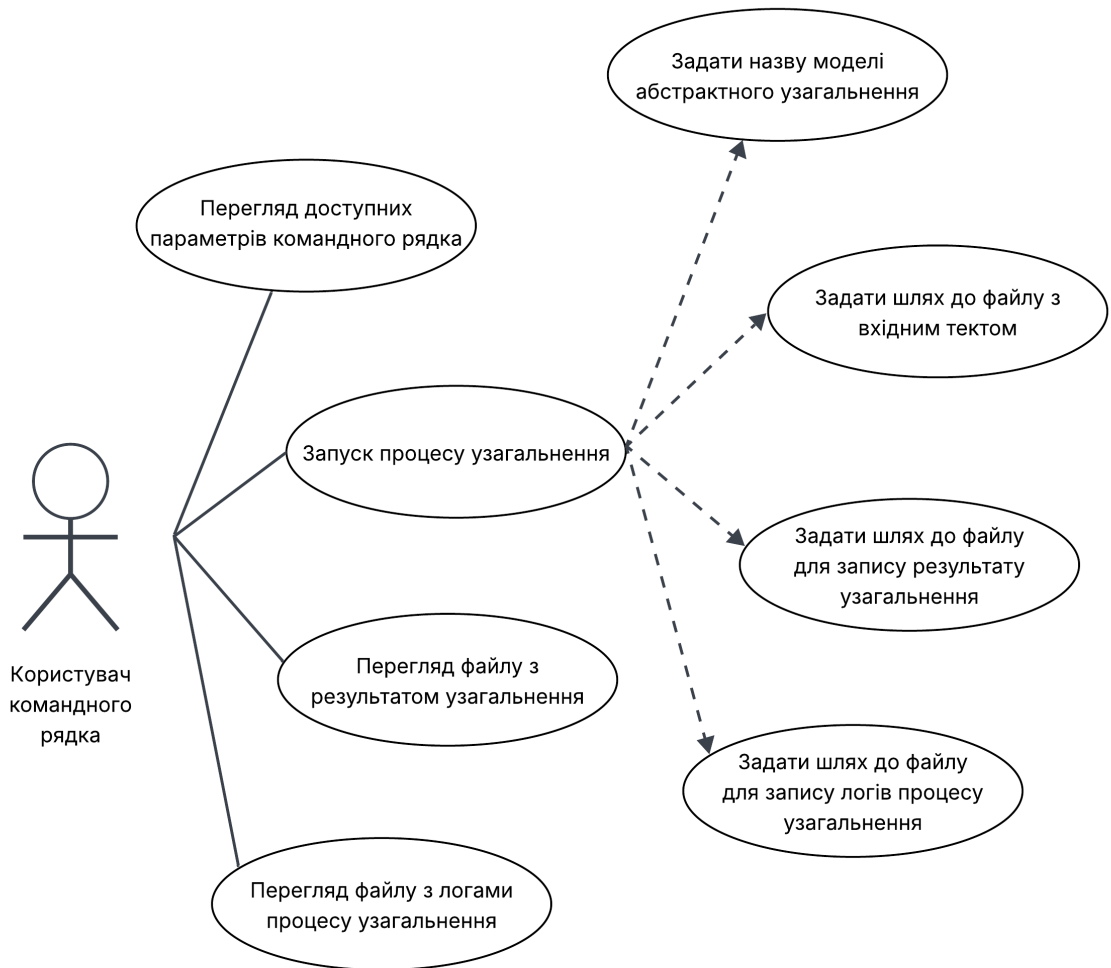


Рис. 3.2. UML діаграма сценаріїв використання користувачем розробленого програмного забезпечення для узагальнення великорозмірних науково-популярних текстів

```

C:\Users\Darrii Kinash\projects\diploma\app>py
app.py --llm bart_cnn --file evolution.txt|
  
```

Рис. 3.3. Приклад запуску розробленого програмного забезпечення

Перелік та опис доступних параметрів командного рядка наведено у табл. 3.1.

## Доступні параметри командного рядка

Параметр	Допустимі значення	Опис
--llm	pegasus, bart_cnn, bart_xsum	Обов'язковий параметр для визначення моделі абстрактного узагальнення, що буде використана на кроці абстрактного узагальнення з застосуванням фрагментації
--file	Шлях до файлу у файлової системі, що містить вхідний текст	Обов'язковий параметр для отримання файлу, що містить вхідний текст
--log	Шлях до файлу, що буде створений для запису логів процесу узагальнення	Необов'язковий параметр, що визначає шлях та назву файлу, що буде створений для запису логів процесу узагальнення. У разі відсутності буде використане значення за замовчуванням
--result	Шлях до файлу, що буде створений для запису результату узагальнення	Необов'язковий параметр, що визначає шлях та назву файлу, що буде створений для запису результату узагальнення. У разі відсутності буде використане значення за замовчуванням

Після завершення роботи додатку буде створено два файли – файл з результатом узагальнення та файл з логами процесу узагальнення – за визначеними користувачем шляхами файлової системи, або ж за шляхами за замовчуванням. Обидва файли будуть створені з розширенням .txt та будуть закодовані у форматі UTF-8. Файл з результатом міститиме текст – результат узагальнення, де кожен абзац відображає окремий тематичний блок – результат узагальнення кластера, тоді як файл з логами міститиме наступні дані:

- назва файлу з вхідним текстом;
- назва обраної моделі абстрактного узагальнення;

- загальна кількість токенів у вхідному тексті;
- швидкість процесу узагальнення (токени на секунду);
- кількість отриманих кластерів;
- відсоток речень з вхідного тексту, що потрапили у кластери;
- тривалість попередньої обробки;
- тривалість обчислення векторних відображень;
- тривалість кластеризації;
- тривалість процесу абстрактного узагальнення з застосуванням фрагментації;
- тривалість сортування;
- загальна тривалість.

Окрім того файл логів міститиме детальну інформацію про кожен кластер:

- кількість речень у кластері;
- кількість токенів у кластері;
- позиція результату узагальнення кластера в кінцевому результаті узагальнення;
- кількість фрагментів, на які був поділений кластер;
- тривалість процесу абстрактного узагальнення кластера з застосуванням фрагментації;
- згенерований результат узагальнення для кластера;
- список речень, що належать до кластера, з інформацією про їхню ймовірність приналежності, кількість токенів та позицію у вхідному тексті.

Нижче наведено приклад файлу логів процесу узагальнення (рис. 3.4) та приклад файлу з результатом узагальнення (рис. 3.5).

```

File:                evolution
LLM:                bart_cnn
Token count:        165644
Speed (tokens/s):   674.4951851820618
Cluster count:      39
Total sentences:     6866
Clustered sentences: 931
Clustered ( % ):    14
Preprocessing time: 0:00:01.381292
Embedding time:     0:00:41.448406
Clustering time:    0:00:34.863303
LLM time:           0:02:47.889182
Sorting time:       0:00:00
Total time:         0:04:05.582183

```

probability ; tokens ; original index

Cluster 1 ( 12 sentences; 218 tokens; 96 position; 1 chunks; 0:00:03.271141 exec time)

Biologists label organisms with a two-part Latin name, genus followed by species. Animals share a common ancestor and share many physical traits.

(0.99; 13; 6)	The story of atoms, molecules and their interactions is called physics.
( 1.0; 8; 8)	The story of organisms is called biology.
( 1.0; 7; 23)	Biologists classify organisms into species.
( 1.0; 25; 24)	Animals are said to belong to the same species if they tend to mate with each other, giving birth to fertile offspring.
( 1.0; 16; 25)	Horses and donkeys have a recent common ancestor and share many physical traits.
(0.98; 20; 27)	They will mate if induced to do so – but their offspring, Neanderthals, are not considered a separate species.
( 1.0; 25; 32)	Species that evolved from a common ancestor are bunched together in a phylogenetic tree.
( 1.0; 17; 34)	Biologists label organisms with a two-part Latin name, genus followed by species.
( 1.0; 14; 241)	Clearly, they were not completely different species like Neanderthals.
( 1.0; 24; 242)	On the other hand, they were not just different populations of the same species.
( 1.0; 37; 245)	Every two species that evolved from a common ancestor, such as horses and donkeys, share a recent common ancestor.
( 1.0; 12; 249)	They were almost, but not quite, entirely separate species.

Cluster 2 ( 28 sentences; 651 tokens; 727 position; 1 chunks; 0:00:04.890554 exec time)

The legend of Peugeot affords us a good example. In what sense can we say that Peugeot SA (the company's official name) exists? There are many Peugeots, but these are obviously not the company. Even if every Peugeot in the world were simultaneously junked and sold for scrap metal, Peugeot SA would not disappear.

(0.91; 20; 386)	The Legend of Peugeot Our chimpanzee cousins usually live in groups.
( 1.0; 13; 444)	The legend of Peugeot affords us a good example.

Рис. 3.4. Приклад створеного файлу з логами процесу узагальнення

```

Biologists label organisms with a two-part Latin name, genus followed by species. Animals are said to belong to the same species if they tend to mate with each other, giving birth to fertile offspring. Horses and donkeys have a recent common ancestor and share many physical traits.

The legend of Peugeot affords us a good example. In what sense can we say that Peugeot SA (the company's official name) exists? There are many Peugeots, but these are obviously not the company. Even if every Peugeot in the world were simultaneously junked and sold for scrap metal, Peugeot SA would not disappear.

Chimps can't win an argument with a Homo sapiens, but the ape can rip the man apart like a rag doll. The alpha male usually wins his position not because he is physically stronger, but because he leads a large and stable coalition.

Humans first evolved in East Africa about 2.5 million years ago from an earlier genus of apes called Australopithecus. Humans in Europe and western Asia evolved into Homo neanderthalensis. Homo erectus, 'Upright Man', survived there for close to 2 million years. According to this view, Sapiens replaced all the previous human populations without merging with them. If that is the case, the lineages of all contemporary humans can be traced back, exclusively, to East Africa, 70,000 years ago. But if the Interbreeding Theory is right, there might well be genetic differences between Africans, Europeans and Asians that go back hundreds of thousands of years. Homo erectus did not undergo further genetic alterations, its stone tools remained roughly the same -- for close to 2 million years. In a one-on-one brawl, a Neanderthal would probably have beaten a Sapiens. But in a conflict of hundreds, Neanderthals wouldn't stand a chance. Homo sapiens evolved to think of people as divided into us and them. Evolution has made Homo sapiens, like other social mammals, a xenophobic creature. The Nazis explained that Homo sapien itself appeared when one 'superior' population of ancient humans evolved.

If a superior man should blind the eye of another superior 197.5 If a superior man strikes a woman of superior class and thereby causes her to miscarry her fetus, he shall weigh and deliver ten shekels of silver for her fetus. If that woman should die, they shall kill his daughter.

We believe in a particular order not because it is objectively true, but because believing in it enables us to cooperate effectively and forge a better society. An imagined order cannot be sustained by violence alone. The imagined order is inter-subjective. In order to change them we must simultaneously change the consciousness of billions of people.

```

Рис. 3.5. Приклад створеного файлу з результатом узагальнення

### 3.5. Висновки до третього розділу

У даному розділі обґрунтовано реалізацію програмного забезпечення у формі консольного додатку з можливістю керування через інтерфейс командного рядка. Також було розглянуто використані засоби для реалізації програмного забезпечення, а саме:

- мова програмування Python;
- бібліотека NLTK;
- бібліотека TensorFlow та платформа TensorFlow Hub;
- бібліотека для кластеризації HDBSCAN;
- бібліотека HuggingFace Transformers.

Окрім того було наведено схему модулів розробленого програмного забезпечення та описано їх призначення та особливості.

Також було виділено особливості програмної реалізації методу узагальнення великорозмірних науково-популярних текстів, а саме:

- налаштування токенизатора бібліотеки NLTK для правильного розбиття тексту на речення;
- обчислення векторних відображень моделлю USE партіями для запобігання перенавантаженню пам'яті пристрою;
- використання бібліотеки HDBSCAN та однойменного алгоритму кластеризації для групування тематичних блоків та суттєвого зменшення розміру вхідного тексту і загальної тривалості виконання програмного забезпечення;
- можливість застосування трьох різних моделей машинного навчання на кроці абстрактного узагальнення з застосуванням фрагментації, що забезпечує різні якісні характеристики результату узагальнення.

На завершення було розглянуто сценарії використання розробленого програмного забезпечення, доступні параметри командного рядка, структуру та приклади файлів з результатом узагальнення та логами процесу узагальнення.

## **4. АНАЛІЗ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ ПРОПОНОВАНОГО МЕТОДУ**

### **4.1. Критерії оцінювання та аналіз ефективності пропонуваного методу**

Постановивши задачу, а саме розробку методу узагальнення, що поєднує основні якісні характеристики результату узагальнення для сприйняття читачем та швидкість і ефективність процесу узагальнення для забезпечення можливості локального виконання, та визначивши еталонний метод для порівняння, а саме багатокроковий метод абстрактного узагальнення з застосуванням фрагментації визначеної контекстом, маємо наступні підзавдання:

- досягти якісних характеристик результату узагальнення, співставних еталонному методу;
- збільшити швидкість виконання процесу узагальнення порівняно з еталонним методом.

При виборі метрик для визначення кількісних характеристик ефективності пропонуваного методу було визначено, що оцінювання якісних характеристик результату узагальнення має бути проведеною вручну, адже наявні кількісні метрики для оцінювання якості результату узагальнення мають значні недоліки при застосуванні їх до результатів узагальнення великорозмірних текстів, тому ефективність пропонуваного методу буде визначена за допомогою оцінювання швидкості виконання пропонуваного методу порівняно з еталонним методом.

Було визначено, що швидкість виконання має обчислюватися як загальна кількість токенів у вхідному тексті поділена на загальний час виконання методу, та вимірюватися в токенах на секунду. Еталонний метод для порівняння був спрощений (рис. 2.3) для полегшення його розробки, що не матиме негативного впливу на оцінювання швидкості виконання пропонуваного методу.

Оскільки програмні реалізації пропонованого та спрощеного еталонного методів надають можливість використання трьох різних моделей абстрактного узагальнення, що мають різну швидкість виконання, а покращена швидкість пропонованого методу досягається за рахунок зменшення кількості викликів моделі, для об'єктивного оцінювання ефективності буде проведено окреме порівняння швидкості для кожної з моделей, а саме *pegasus-large*, *bart-large-cnn* та *bart-large-xsum*.

Для оцінювання ефективності з публічно доступних онлайн бібліотек було взято 50 екземплярів великорозмірних науково-популярних текстів, 5 з яких, інформація про які наведена нижче, були використані для ручної оцінки якості результату узагальнення:

1. “*Sapiens: A Brief History of Humankind*” авторства Ювала Ноя Харарі [47], що містить 6866 речень та 165644 токени моделі *bart-large-cnn*.
2. “*A Brief History of Time*” авторства Стівена Гокінга [48], що містить 2527 речень та 73396 токенів моделі *bart-large-cnn*.
3. “*The Elegant Universe*” авторства Браяна Гріна [49], що містить 5327 речень та 175429 токенів моделі *bart-large-cnn*.
4. “*The Uninhabitable Earth: Life After Warming*” авторства Девіда Воллес-Веллса [50], що містить 2228 речень та 89355 токенів моделі *bart-large-cnn*.
5. “*Thinking, Fast and Slow*” авторства Деніела Канемана [51], що містить 7098 речень та 195737 токенів моделі *bart-large-cnn*.

Отримані значення швидкості роботи для пропонованого та спрощеного еталонного методів для кожної доступної моделі, а також значення відсотка кластеризованих речень, наведені у табл. 4.1, де значення швидкості роботи вимірюються у токенах на секунду.

Таблиця 4.1

Значення швидкості роботи пропонованого та еталонного методів

Назва тексту	Відсоток кластеризації	pegasus-large		bart-large-cnn		bart-large-xsum	
		Пропонований	Еталонний	Пропонований	Еталонний	Пропонований	Еталонний
Sapiens: A Brief History of Humankind	14	159	32	675	337	930	536
A Brief History of Time	10	251	37	1026	351	1300	519
The Elegant Universe	6	399	41	1388	386	1775	496
The Uninhabitable Earth: Life After Warming	5	331	38	1248	359	1470	553
Thinking, Fast and Slow	10	217	37	889	346	1117	525
Округлене середнє арифметичне		271	37	1045	356	1318	526

Як бачимо, у всіх випадках пропонований метод швидший за спрощений еталонний. Також примітно те, що тоді як в спрощеного еталонного методу швидкість мало відрізняється при обробці різних текстів, у пропонованого методу вона залежить від відсотка кластеризованих речень, адже в різних текстах ключові теми можуть різнитися кількістю та мірою своєї присутності в тексті, відповідно буде виконана менша чи більша

кількість викликів до моделі абстрактного узагальнення, що й вплине на швидкість роботи пропонованого методу. Окрім того можна звернути увагу на різницю швидкості роботи самих моделей машинного навчання. Як бачимо, модель *bart-large-xsum* трохи швидша за модель *bart-large-cnn*, тоді як модель *pegasus-large* має найнижчу швидкість зі значним відривом. Це відбувається через різницю архітектур моделей та підходів до їх тренування, адже модель *bart-large-xsum* тренується на максимально коротких стислих викладах, тоді як модель *pegasus-large* має додаткове завдання маскуванню цілих речень під час тренування. Зведення середніх швидкостей пропонованого та спрощеного еталонного методів, а також “найгіршої” швидкості пропонованого методу, де відсоток кластеризації найбільший, для *pegasus-large* (рис. 4.1), *bart-large-cnn* (рис. 4.2) та *bart-large-xsum* (рис. 4.3) зображено нижче.

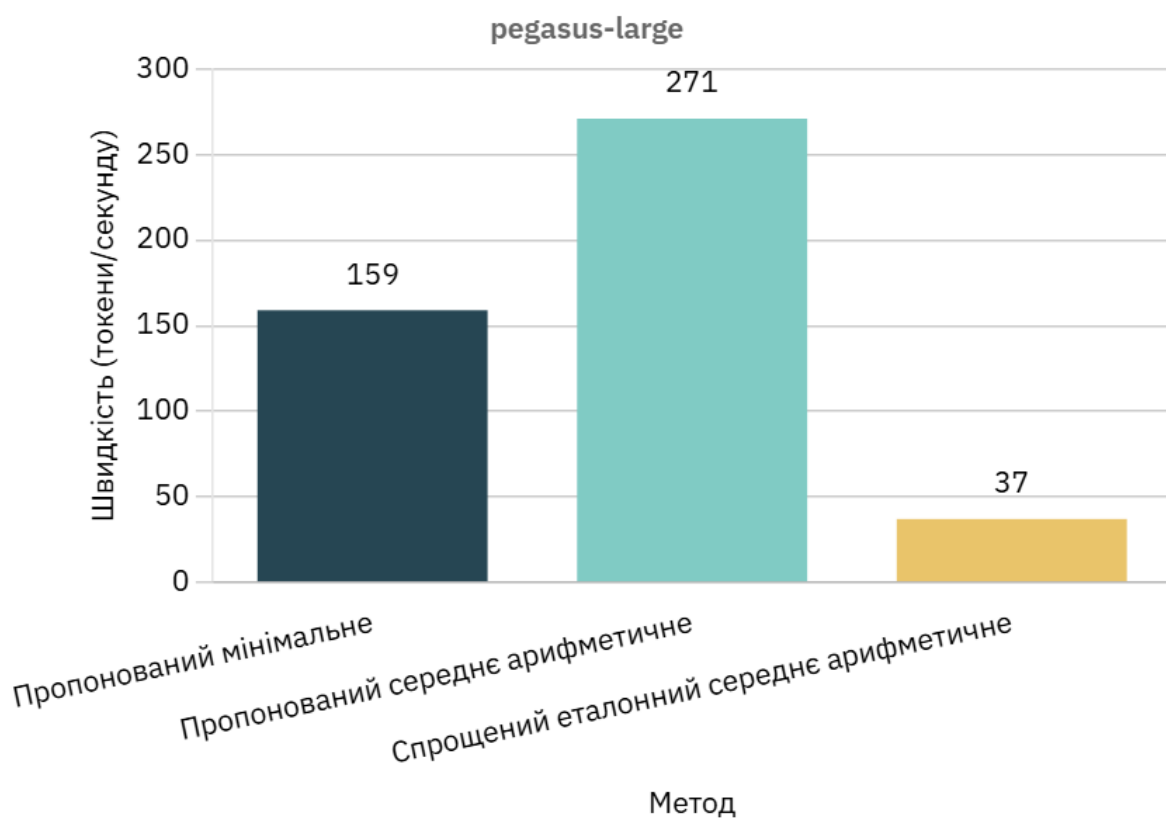


Рис. 4.1. Зведене порівняння швидкості виконання для *pegasus-large*

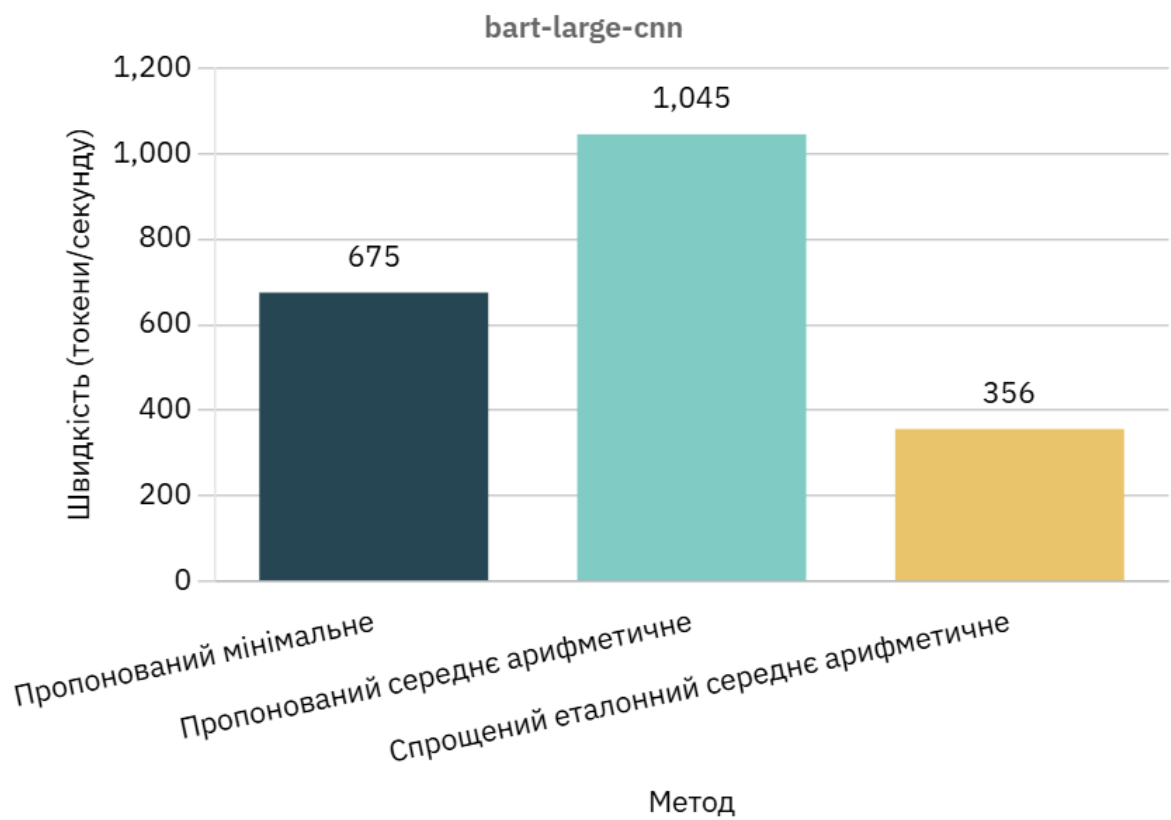


Рис. 4.2. Зведене порівняння швидкості виконання для bart-large-cnn

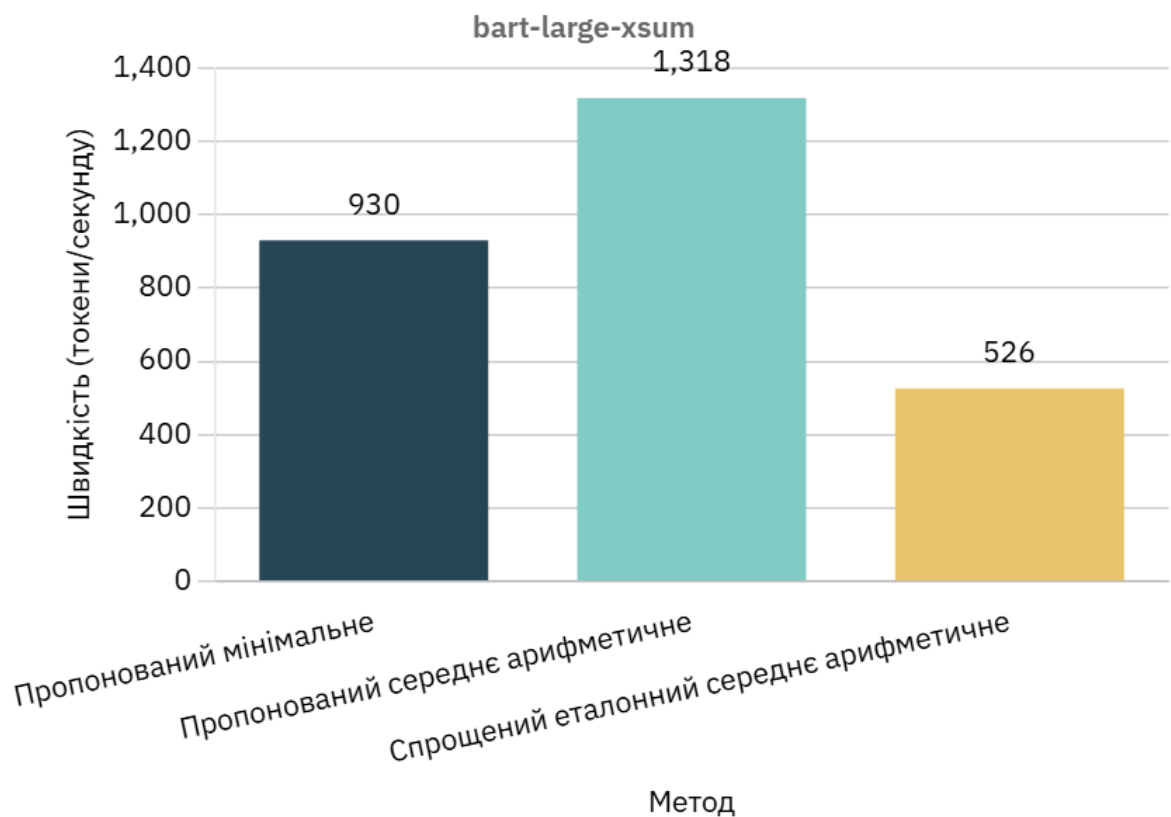


Рис. 4.3. Зведене порівняння швидкості виконання для bart-large-xsum

Порівнявши мінімальне значення швидкості запропонованого методу з середнім арифметичним значенням швидкості спрощеного еталонного методу бачимо, що запропонований метод як мінімум в 4.3, 1.9 та 1.8 разів швидший за спрощений еталонний метод при використанні реалізації моделі pegasus-large, bart-large-cnn та bart-large-xsum відповідно в обох методах. Як вже було зазначено раніше, велика різниця між результатами для моделей PEGASUS та BART спричинена тим, що реалізація моделі PEGASUS більш ресурсозатратна та обробляє текст набагато довше ніж реалізації моделі BART.

#### 4.2. Огляд отриманих результатів

Проведемо огляд процесу узагальнення на прикладі тексту “Sapiens: A Brief History of Humankind”. Текст містить 6866 речень, що становлять 165644 токенів моделі BART або 161145 токенів моделі PEGASUS. На кроці кластеризації було виділено 39 кластерів, а саме 931 речення з 6866, що складає близько 14% від загальної кількості речень. Для прикладу розглянемо кластер №20, представлений в табл. 4.2, де ймовірність – це ймовірність приналежності речення до кластера, а позиція – це початкова позиція речення у вхідному тексті для забезпечення правильної послідовності речень.

Таблиця 4.2

Кластер виділений з тексту “Sapiens: A Brief History of Humankind”

Ймовірність	Позиція	Речення
1	2818	In order to understand the limitations of barter, imagine that you own an apple orchard in the hill country that produces the crispest, sweetest apples in the entire province.
1	2822	You find the shoemaker's shop and offer to barter some of your apples in exchange for the shoes you need.

1	2826	The last time the shoemaker exchanged shoes for apples was three months ago, and back then he asked for three sacks of apples.
0.98	2828	On the other hand, on that previous occasion, the apples were given in exchange for small women's shoes.
1	2833	In a barter economy, every day the shoemaker and the apple grower will have to learn anew the relative prices of dozens of commodities.
0.98	2834	If one hundred different commodities are traded in the market, then buyers and sellers will have to know 4,950 different exchange rates.
1	2836	Even if you manage to calculate how many apples equal one pair of shoes, barter is not always possible.
1	2838	What happens if the shoemaker doesn't like apples and, if at the moment in question, what he really wants is a divorce?
1	2840	But what if the lawyer is full up on apples but really needs a haircut?
1	2869	A shoemaker in a money economy needs to know only the prices charged for various kinds of shoes – there is no need to memorise the exchange rates between shoes and apples or goats.
1	2873	The shoemaker will always be happy to take your money, because no matter what he really wants – apples, goats or a divorce – he can get it in exchange for money.
0.99	2953	The mark imprinted on them testifies to their exact value, so the shoemaker doesn't have to keep a scale on his cash register.
1	5368	Yet what happens if the greedy shoemaker increases his profits by paying employees less and increasing their work hours?
1	5370	If our shoemaker pays too little and demands too much, the best employees would naturally abandon him and go to work for his competitors.
1	5371	The tyrant shoemaker would find himself left with the worst labourers, or with no labourers at all.

1	5376	If there is a single corporation controlling all shoe factories in a country, or if all factory owners conspire to reduce wages simultaneously, then the labourers are no longer able to protect themselves by switching jobs.
1	5377	Even worse, greedy bosses might curtail the workers' freedom of movement through debt peonage or slavery.

Як бачимо, даний кластер був виділений на основі тематичного блоку, що пояснює принцип бартерної економіки та важливість валюти на прикладі садівника та шевця. Окрім того, як можна зрозуміти, проаналізувавши позиції, в кластер були включені речення з іншого тематичного блоку, що пояснює проблеми жадібності корпорацій та експлуатації робітників. Хоч за суттю ці два тематичні блоки не пов'язані, вони сформульовані за допомогою прикладу про шевця, тому були згруповані разом. На даний момент такий випадок могли б обробити лише більш просунуті та набагато ресурсоемніші моделі машинного навчання, використання яких би унеможливило можливість локального виконання на пристроях користувача.

Загалом можна зазначити, що кластер відповідає одній тематиці та зберігає початкову послідовність речень, тож можна вважати крок екстрактного узагальнення з застосуванням кластеризації успішним. Генерація векторних відображень зайняла 40 секунд, тоді як сама кластеризація зайняла 35 секунд, що дає загальний час 1 хвилину 15 секунд для кроку екстрактного узагальнення з застосуванням кластеризації.

Тепер розглянемо другий крок – крок абстрактного узагальнення з застосуванням фрагментації визначеної вмістом – для кожної доступної реалізації моделей абстрактного узагальнення. Оскільки загальна кількість токенів для кластера (462 для моделі BART та 454 для моделі PEGASUS) менша за обмеження в 1024 токени для моделі, абстрактне узагальнення проводиться без фрагментації. Розглянемо результати абстрактного

узагальнення кластера для кожної реалізації моделей BART та PEGASUS (табл. 4.3).

Таблиця 4.3

Результати узагальнення кластера

Назва реалізації моделі	Час обробки в секундах	Результат узагальнення
pegasus-large	17	A shoemaker in a money economy needs to know only the prices charged for various kinds of shoes – there is no need to memorise the exchange rates between shoes and apples or goats. The shoemaker will always be happy to take your money, because no matter what he really wants – apples, goats or a divorce – he can get it in exchange for money.
bart-large-cnn	5	In a barter economy, every day the shoemaker and the apple grower will have to learn anew the relative prices of dozens of commodities. Even if you manage to calculate how many apples equal one pair of shoes, barter is not always possible. A shoemaker in a money economy needs to know only the prices charged for various kinds of shoes.
bart-large-xsum	3	In our series of letters from Asian journalists, Chinese writer and film-maker Sun Yat-sen looks at the limits of the barter economy.

Як бачимо, результати узагальнення моделями pegasus-large та bart-large-cnn є зв'язними та послідовними, а результат bart-large-cnn трохи краще відображає суть кластера та виконує обробку тексту набагато швидше за pegasus-large. Натомість результат bart-large-xsum, хоч і є найкоротшим та найшвидше обчисленим, демонструє значну проблему галюцинації мовної моделі, що може виникати на етапі навчання через неправильно підібрані дані або метод тренування [52], адже фрази “series of letters from Asian journalists” та “Chinese writer and film-maker Sun Yat-sen” чи будь-які їх підпоследовності та комбінації відсутні у реченнях кластера,

тому необхідним є використання правильно натренованої моделі абстрактного узагальнення.

Тривалість виконання кроку абстрактного узагальнення з застосуванням фрагментації становить 15 хвилин 32 секунди, 2 хвилини 50 секунд та 1 хвилину 45 секунд для реалізацій моделей `pegasus-large`, `bart-large-cnn` та `bart-large-xsum` відповідно.

Наступним кроком всі результати узагальнення кластерів конкатенуються у кінцеве узагальнення у порядку, визначеному середнім арифметичним позицій речень кожного кластера, чим і досягається послідовність тематичних блоків у кінцевому узагальненні. Загальна тривалість виконання пропонованого методу становить 16 хвилин 55 секунд, 4 хвилини 5 секунд та 3 хвилини 0 секунд для реалізацій моделей `pegasus-large`, `bart-large-cnn` та `bart-large-xsum` відповідно. Тоді як загальна тривалість виконання спрощеного еталонного методу становить 1 годину 25 хвилин, 8 хвилин 10 секунд та 5 хвилин 10 секунд для реалізацій моделей `pegasus-large`, `bart-large-cnn` та `bart-large-xsum` відповідно.

Наостанок було проведено ручне оцінювання якості результату узагальнення пропонованого методу порівняно з результатом узагальнення еталонного методу. Було визначено, що зв'язність результатів узагальнення обох методів знаходиться на однаковому рівні. Охоплення тем тексту в результаті узагальнення пропонованого методу залежить від виразності їх висвітлення у вхідному тексті, тоді як в результаті узагальнення еталонного методу воно залежить від кількості виконаних ітерацій. В цілому при виконанні одної ітерації еталонного методу його результат охоплює більше тематичних блоків, зокрема більш широко та абстрактно описаних у вхідному тексті, та має задовільну послідовність викладу думки, однак в такому випадку результат є недостатньо стислим. У випадку виконання двох ітерацій результат стає набагато більш стислим, а також втрачає багато тематичних блоків та послідовність викладу думки для кожного блоку. Натомість результат узагальнення пропонованого методу зберігає більшість

тематичних блоків, хоч і не в такій мірі, як еталонний метод, зберігає задовільну послідовність викладу думки в кожному блоці, а також є достатньо стислим.

Отже, запропонований метод досягає значного підвищення швидкості роботи при мінімальному зниженні якості результату узагальнення порівняно з еталонним методом.

### **4.3. Напрямки подальшої роботи**

Напрямки подальшої роботи над покращенням запропонованого методу узагальнення великорозмірних науково-популярних текстів полягають у розширенні його можливостей та виправленні потенційних недоліків. Перспективними вважаються наступні способи покращення методу:

- Повноцінне тренування наявних моделей узагальнення на спеціально підготовлених даних для запобігання проблеми галюцинації.
- Додавання можливості паралелізованого виконання моделей машинного навчання на пристроях з наявним графічним процесором для подальшого збільшення швидкості роботи методу.
- Додаткова обробка векторних відображень речень або заміна моделі машинного навчання, що обчислює векторні відображення, на таку, що повертає вектори меншої вимірності, для зменшення впливу “прокляття високимірності” на процес кластеризації.
- Перевірка інших видів великорозмірних текстів на сумісність з запропонованим методом.
- Доопрацювання коду для можливості застосування методу в якості програмної бібліотеки.

#### **4.4. Висновки до розділу**

У даному розділі було проаналізовано ефективність розробленого програмного забезпечення, що реалізує запропонований метод узагальнення великорозмірних науково-популярних текстів. Оцінювання ефективності методу було проведено за критерієм швидкості роботи в порівнянні з спрощеним багатокроковим методом абстрактного узагальнення з застосуванням фрагментації визначеної контекстом. Окрім того було проведено ручне оцінювання якості результату узагальнення в порівнянні з результатом узагальнення еталонного методу. В результаті оцінювання було з'ясовано, що запропонований метод має кращі показники швидкості роботи при мінімальній втраті якості результату узагальнення.

Окрім того було наведено напрямки подальшої роботи, а саме повноцінне тренування використаних моделей, забезпечення можливості виконання на графічному процесорі, зменшення вимірності векторних відображень речень для покращення кластеризації, перевірка інших видів великорозмірних текстів на сумісність з запропонованим методом та доопрацювання коду для можливості застосування методу в якості програмної бібліотеки.

## ВИСНОВКИ

Дана дисертація присвячена створенню методу та програмного забезпечення узагальнення великорозмірних науково-популярних текстів.

В ході дослідження розглянуто поняття, мету та види процесу узагальнення, визначено поняття великорозмірного тексту, розглянуто поняття науково-популярної літератури, визначено, чому вона найкраще підходить для створення методу узагальнення. Окрім того було розглянуто наявні методи та підходи узагальнення великорозмірних текстів та проаналізовано їх переваги та недоліки. Також було розглянуто тенденції щодо застосування кордонних обчислень у сфері ШІ і локального запуску моделей машинного навчання та зроблено висновок про необхідність розроблення методу узагальнення великорозмірних науково-популярних текстів, що буде мати покращену швидкість роботи без втрати якісних характеристик результату узагальнення порівняно з найкращим наявним методом.

Для забезпечення покращеної швидкості роботи без втрати якісних характеристик результату узагальнення при проектуванні методу було вирішено застосувати двокроковий підхід, де першим кроком застосовується екстрактне узагальнення з застосуванням кластеризації для швидкого зменшення розміру вхідного тексту і збереження логічної послідовності шляхом виділення основних тематичних блоків, а другим кроком застосовується абстрактне узагальнення з застосуванням фрагментації визначеної вмістом для забезпечення подальшого зменшення розміру тексту та збереження якісних характеристик результату узагальнення. Було детально описано етапи та алгоритм пропонованого методу, а також визначено та описано метод для порівняння при оцінюванні ефективності – спрощений багатокроковий метод абстрактного узагальнення з застосуванням фрагментації визначеної контекстом. Окрім

того було розглянуто метрики для оцінювання ефективності пропонованого методу.

Було проведено розробку програмної реалізації пропонованого методу на мові програмування Python у вигляді консольного додатку з можливістю керування за допомогою інтерфейсу командного рядка. На додачу було описано використані лінгвістичні пакети та бібліотеки машинного навчання, розглянуто функції модулів розробленого програмного забезпечення, описано архітектуру та особливості реалізації додатку та наведено приклад використання.

Проведено аналіз ефективності запропонованого методу за критерієм швидкості роботи та наведено обґрунтування отриманих результатів. Порівняння з еталонним методом показує, що запропонований метод дозволяє підвищити швидкість обробки тексту як мінімум в 1.8 разів. При ручному оцінюванні якості результату узагальнення було визначено, що у запропонованого методу в порівнянні з еталонним методом присутні незначні втрати якості узагальнення, а саме втрата широко визначних та абстрактних тематичних блоків.

Подальші напрямки роботи над удосконаленням запропонованого методу включають повноцінне тренування використаних моделей, забезпечення можливості виконання на графічному процесорі, зменшення вимірності векторних відображень речень для покращення кластеризації, перевірка інших видів великорозмірних текстів на сумісність з запропонованим методом та доопрацювання коду для можливості застосування методу в якості програмної бібліотеки.

## СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. I. Mani, "Automatic Summarization", John Benjamins Publishing, 2001.
2. A. Nenkova, and K. McKeown, "A Survey of Text Summarization Techniques", Mining Text Data, pp. 43-76, 2012.
3. K. Spärck Jones, "A Statistical Interpretation of Term Specificity and its Application in Retrieval", Journal of Documentation, pp. 11-21, 1972.
4. S. Shearing, A. Gertner, B. Wellner, and L. Merkhofer, "Automated Text Summarization: A Review and Recommendations", 2020.
5. I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The Long-Document Transformer", arXiv:2004.05150v2, 2020.
6. О. В. Романенко, "Науково-популярна література", Енциклопедія Сучасної України, [Електронний ресурс]. Режим доступу: <https://esu.com.ua/article-70702>
7. R. Mihalcea, and P. Tarau, "TextRank: Bringing Order into Text", Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 404-411, 2004.
8. G. Erkan, and D. Radev, "LexRank: Graph-based Lexical Centrality as Saliency in Text Summarization", Journal Of Artificial Intelligence Research, Vol. 22, pp. 457-479, 2004.
9. L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web", [Електронний ресурс]. Режим доступу: <http://ilpubs.stanford.edu:8090/422>
10. A. Fabbri, I. Li, T. She, S. Li, and D. Radev, "Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model", Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 1074-1084, 2019.
11. A. Kus, and C. Aci, "A Hybrid Approach to Automatic Text Summarization of Turkish Texts: Integrating Extractive Methods with LLMs", 2024

- Innovations in Intelligent Systems and Applications Conference, pp. 1-6, 2024.
12. A. Sharaff, M. Jain, and G. Modugula, “Feature based cluster ranking approach for single document summarization”, *International Journal of Information Technology*, Vol. 14, pp. 2057-2065, 2022.
  13. Content Chunking: A Foundation of Effective AI in Knowledge Work, [Электронный ресурс]. Режим доступа: <https://www.ais.com/content-chunking-a-foundation-of-effective-ai-in-knowledge-work>
  14. P. Wilman, T. Atara, and D. Suhartono, “Abstractive English Document Summarization Using BART Model with Chunk Method”, *Procedia Computer Science*, Vol 245, pp. 1010-1019, 2024.
  15. C. Leiter, R. Zhang, Y. Chen, J. Belouadi, D. Larionov, V. Fresen, and S. Eger, “ChatGPT: A meta-analysis after 2.5 months”, *Machine Learning with Applications*, Vol 16, 2024.
  16. A. Shaji George, A. Hovan George, and T. Baskar, “Edge Computing and the Future of Cloud Computing: A Survey of Industry Perspectives and Predictions”, *Partners Universal International Research Journal*, Vol. 2, pp. 19-44, 2023.
  17. Apple Core ML, [Электронный ресурс]. Режим доступа: <https://developer.apple.com/machine-learning/core-ml/>
  18. Samsung Edge AI, [Электронный ресурс]. Режим доступа: <https://semiconductor.samsung.com/news-events/tech-blog/on-the-edge-how-edge-ai-is-reshaping-the-future/>
  19. Amazon GreenGrass, [Электронный ресурс]. Режим доступа: <https://aws.amazon.com/greengrass/features>
  20. W. Dai, and Q. He, “Automatic summarization model based on clustering algorithm”, *Scientific Reports*, Vol. 14, 2024.
  21. J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, *arXiv:1810.04805v2*, 2019.

22. S. Lloyd, "Least squares quantization in PCM", Transactions on Information Theory, Vol. 28, pp. 129-137, 1982.
23. M. Ahmed, R. Seraj, and S. Islam, "The k-means Algorithm: A Comprehensive Survey and Performance Evaluation", Electronics, Vol. 9, pp. 1-12, 2020.
24. I. Khan, H. Daud, N. Zainuddin, R. Sokkalingam, Abdussamad, A. Museeb, and A. Inayat, "Addressing limitations of the K-means clustering algorithm: outliers, non-spherical data, and optimal cluster selection", AIMS Mathematics, Vol. 9, 2024.
25. D. Kinash, T. Zabolotnia, "The combined method for a long text summarization", Applied mathematics and computing AMC' 2024, pp. 104-109, 2024.
26. D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil, "Universal Sentence Encoder", arXiv:1803.11175v2, 2018.
27. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need", Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017.
28. L. McInnes, J. Healy, and S. Astels, "HDBSCAN: Hierarchical density based clustering", The Journal of Open Source Software, 2017.
29. Comparing Python Clustering Algorithms, [Электронный ресурс]. Режим доступа:  
[https://hdbscan.readthedocs.io/en/latest/comparing\\_clustering\\_algorithms.html](https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html)
30. Benchmarking Performance and Scaling of Python Clustering Algorithms, [Электронный ресурс]. Режим доступа:  
[https://hdbscan.readthedocs.io/en/latest/performance\\_and\\_scalability.html](https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html)
31. M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising Sequence-to-Sequence

- Pre-training for Natural Language Generation, Translation, and Comprehension”, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 7871-7880, 2020.
32. J. Zhang, Y. Zhao, M. Saleh, and P. Liu, “PEGASUS: pre-training with extracted gap-sentences for abstractive summarization”, Proceedings of the 37th International Conference on Machine Learning, Vol. 119, pp. 11328-11339, 2020.
  33. C. Lin, “ROUGE: A Package for Automatic Evaluation of Summaries”, Text Summarization Branches Out, pp. 74-81, 2004.
  34. K. Papineni, S. Roukos, T. Ward, and W. Zhu, “Bleu: a Method for Automatic Evaluation of Machine Translation”, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311-318, 2002.
  35. S. Banerjee, and A. Lavie, “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments”, Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, pp. 65-72, 2005.
  36. WordNet: A Lexical Database for English, [Электронный ресурс]. Режим доступа: <https://wordnet.princeton.edu>
  37. T. Zhang, V. Kishore, F. Wu, K. Weinberger, and Y. Artzi, “BERTScore: Evaluating Text Generation with BERT”, arXiv:1904.09675v3, 2020.
  38. N. Schluter, “The limits of automatic summarisation according to ROUGE”, Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Vol. 2, pages 41–45, 2017.
  39. C. Shen, L. Cheng, X. Nguyen, Y. You, and L. Bing, “Large Language Models are Not Yet Human-Level Evaluators for Abstractive Summarization”, Findings of the Association for Computational Linguistics, pp. 4215-4233, 2023.
  40. Natural Language Toolkit, [Электронный ресурс]. Режим доступа: <https://www.nltk.org>

41. TensorFlow, [Электронный ресурс]. Режим доступа:  
<https://www.tensorflow.org>
42. TensorFlow Hub, [Электронный ресурс]. Режим доступа:  
<https://www.tensorflow.org/hub>
43. HuggingFace Transformers, [Электронный ресурс]. Режим доступа:  
<https://huggingface.co/docs/transformers>
44. Pegasus-large, [Электронный ресурс]. Режим доступа:  
<https://huggingface.co/google/pegasus-large>
45. Bart-large-cnn, [Электронный ресурс]. Режим доступа:  
<https://huggingface.co/facebook/bart-large-cnn>
46. Bart-large-xsum, [Электронный ресурс]. Режим доступа:  
<https://huggingface.co/facebook/bart-large-xsum>
47. Y. Harari, “Sapiens: A Brief History of Humankind”, 2015.
48. S. Hawking, “A Brief History of Time”, 1989.
49. B. Greene, “The Elegant Universe”, 1999.
50. D. Wallace-Wells, “The Uninhabitable Earth: Life After Warming”, 2019.
51. D. Kahneman , “Thinking, Fast and Slow”, 2011.
52. L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu, “A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions”, arXiv:2311.05232v2, 2024.

## **ДОДАТКИ**

**Додаток 1**  
**Копії графічних матеріалів**

Схема етапів комбінованого методу узагальнення великорозмірних науково-популярних текстів



# Алгоритм абстрактного узагальнення з застосуванням фрагментації визначеної вмістом

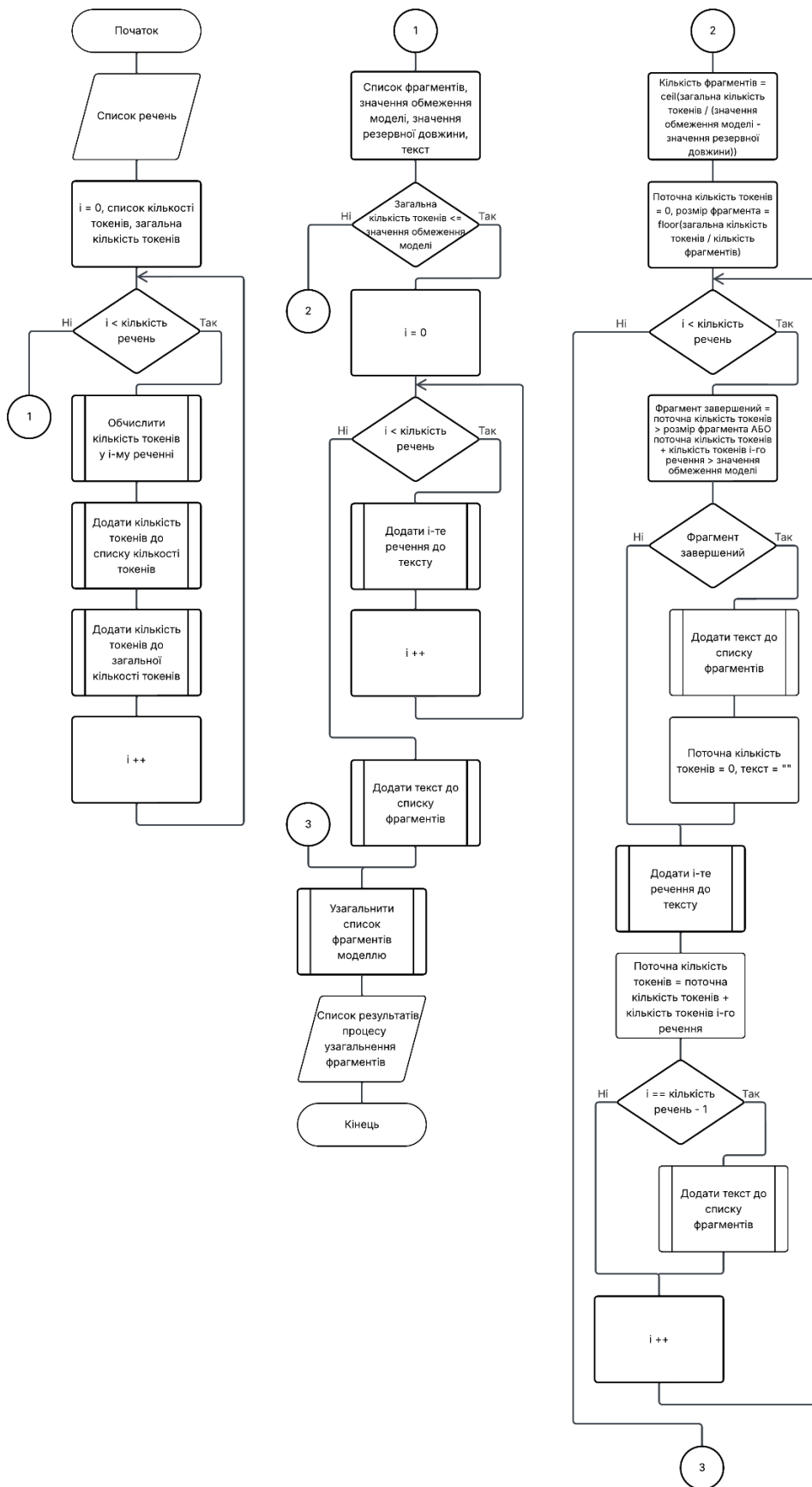
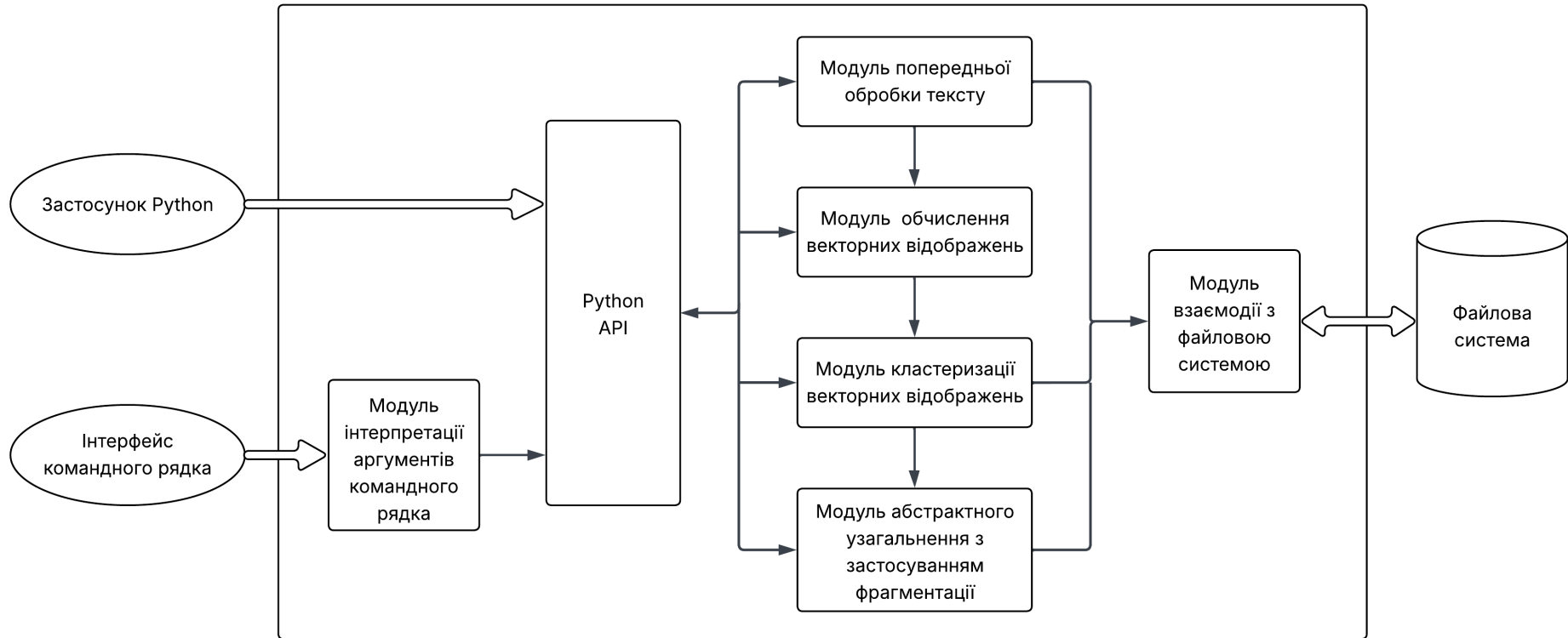


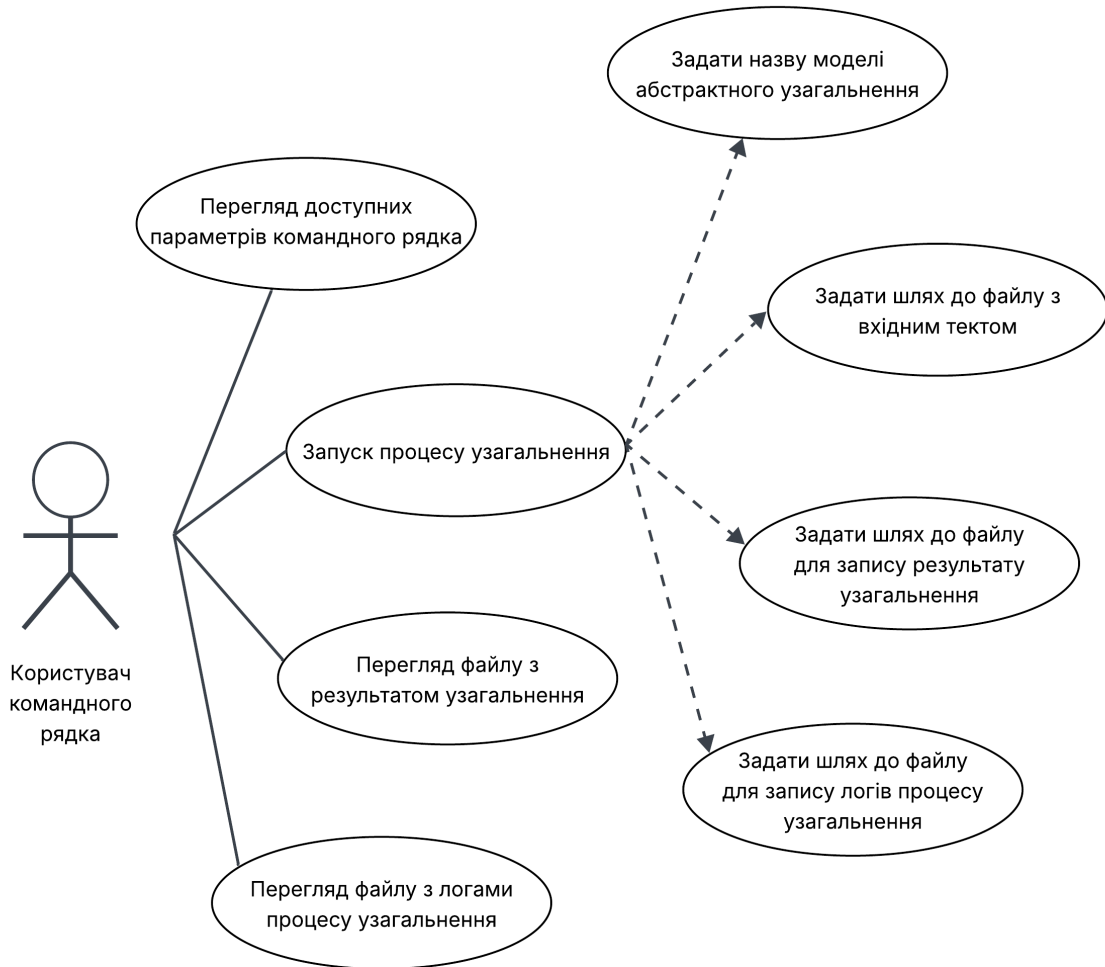
Схема етапів спрощеного методу багатокрокового  
абстрактного узагальнення



Діаграма модулів розробленого програмного забезпечення



UML діаграма сценаріїв використання розробленого програмного  
забезпечення для узагальнення великорозмірних  
науково-популярних текстів



Кінаш Дарій, КП-31мн

## Приклад створеного файлу з логами процесу узагальнення

```
File:                evolution
LLM:                bart_cnn
Token count:        165644
Speed (tokens/s):   674.4951851820618
Cluster count:      39
Total sentences:     6866
Clustered sentences: 931
Clustered ( % ):    14
Preprocessing time: 0:00:01.381292
Embedding time:     0:00:41.448406
Clustering time:    0:00:34.863303
LLM time:           0:02:47.889182
Sorting time:       0:00:00
Total time:         0:04:05.582183

probability ; tokens ; original index

Cluster 1 ( 12 sentences; 218 tokens; 96 position; 1 chunks; 0:00:03.271141 exec time)

Biologists label organisms with a two-part Latin name, genus followed by species. Animals
common ancestor and share many physical traits.

(0.99; 13; 6)      The story of atoms, molecules and their interactions is c
( 1.0; 8; 8)      The story of organisms is called biology.
( 1.0; 7; 23)     Biologists classify organisms into species.
( 1.0; 25; 24)    Animals are said to belong to the same species if they te
( 1.0; 16; 25)    Horses and donkeys have a recent common ancestor and shar
(0.98; 20; 27)    They will mate if induced to do so – but their offspring,
( 1.0; 25; 32)    Species that evolved from a common ancestor are bunched t
( 1.0; 17; 34)    Biologists label organisms with a two-part Latin name, ge
( 1.0; 14; 241)   Clearly, they were not completely different species like
( 1.0; 24; 242)   On the other hand, they were not just different populatio
( 1.0; 37; 245)   Every two species that evolved from a common ancestor, su
( 1.0; 12; 249)   They were almost, but not quite, entirely separate specie

Cluster 2 ( 28 sentences; 651 tokens; 727 position; 1 chunks; 0:00:04.890554 exec time)

The legend of Peugeot affords us a good example. In what sense can we say that Peugeot SA
simultaneously junked and sold for scrap metal, Peugeot SA would not disappear.

(0.91; 20; 386)   The Legend of Peugeot Our chimpanzee cousins usually live
( 1.0; 13; 444)   The legend of Peugeot affords us a good example.
```

Кінаш Дарій, КП-31мн

## Приклад створеного файлу з результатом узагальнення

Biologists label organisms with a two-part Latin name, genus followed by species. Animals are said to belong to the same species if they tend to mate with each other, giving birth to fertile offspring. Horses and donkeys have a recent common ancestor and share many physical traits.

The legend of Peugeot affords us a good example. In what sense can we say that Peugeot SA (the company's official name) exists? There are many Peugeots, but these are obviously not the company. Even if every Peugeot in the world were simultaneously junked and sold for scrap metal, Peugeot SA would not disappear.

Chimps can't win an argument with a Homo sapiens, but the ape can rip the man apart like a rag doll. The alpha male usually wins his position not because he is physically stronger, but because he leads a large and stable coalition.

Humans first evolved in East Africa about 2.5 million years ago from an earlier genus of apes called Australopithecus. Humans in Europe and western Asia evolved into Homo neanderthalensis. Homo erectus, 'Upright Man', survived there for close to 2 million years.

According to this view, Sapiens replaced all the previous human populations without merging with them. If that is the case, the lineages of all contemporary humans can be traced back, exclusively, to East Africa, 70,000 years ago. But if the Interbreeding Theory is right, there might well be genetic differences between Africans, Europeans and Asians that go back hundreds of thousands of years.

Homo erectus did not undergo further genetic alterations, its stone tools remained roughly the same -- for close to 2 million years. In a one-on-one brawl, a Neanderthal would probably have beaten a Sapiens. But in a conflict of hundreds, Neanderthals wouldn't stand a chance.

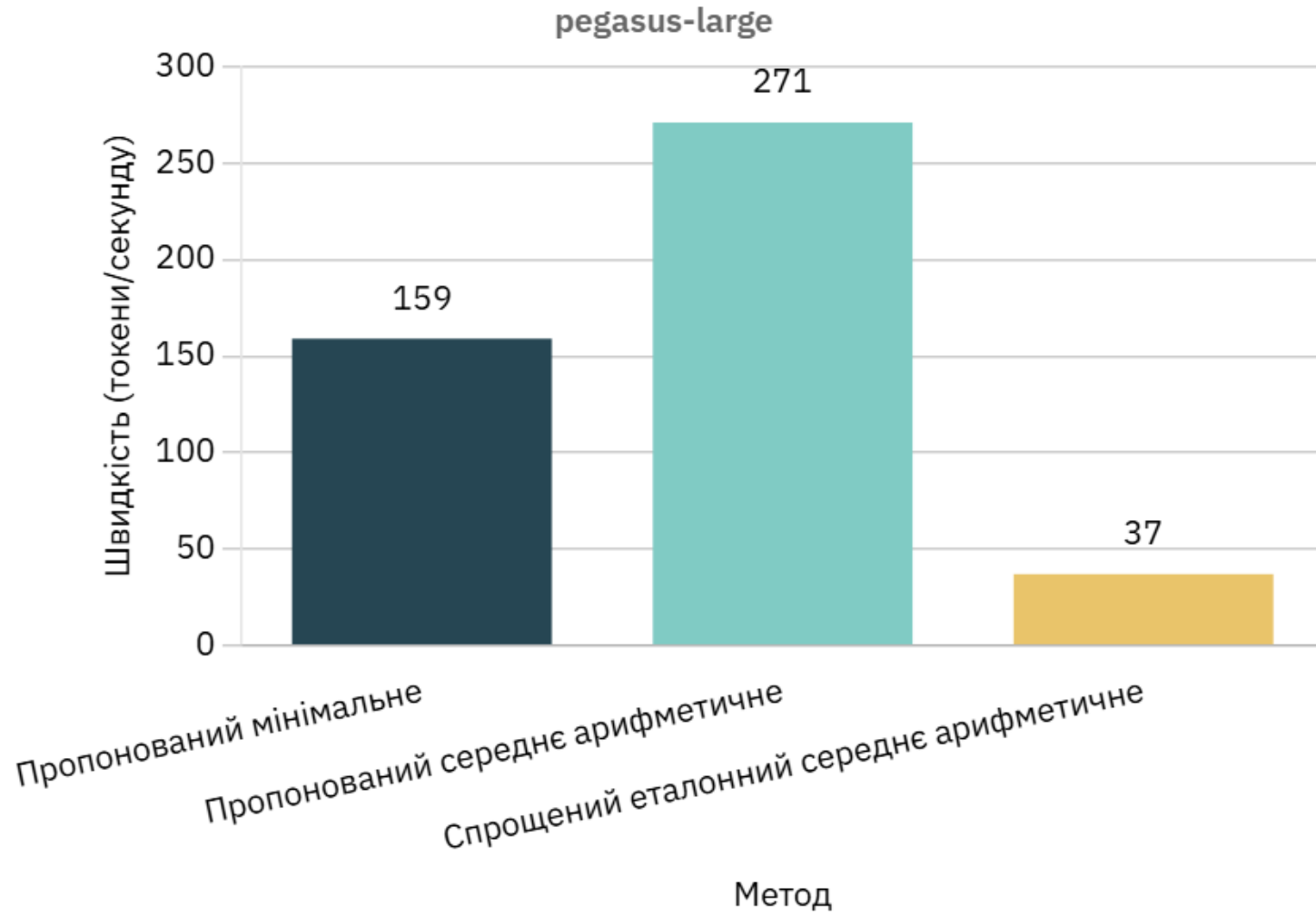
Homo sapiens evolved to think of people as divided into us and them. Evolution has made Homo sapiens, like other social mammals, a xenophobic creature. The Nazis explained that Homo sapiens itself appeared when one 'superior' population of ancient humans evolved.

If a superior man should blind the eye of another superior 197.5 If a superior man strikes a woman of superior class and thereby causes her to miscarry her fetus, he shall weigh and deliver ten shekels of silver for her fetus. If that woman should die, they shall kill his daughter.

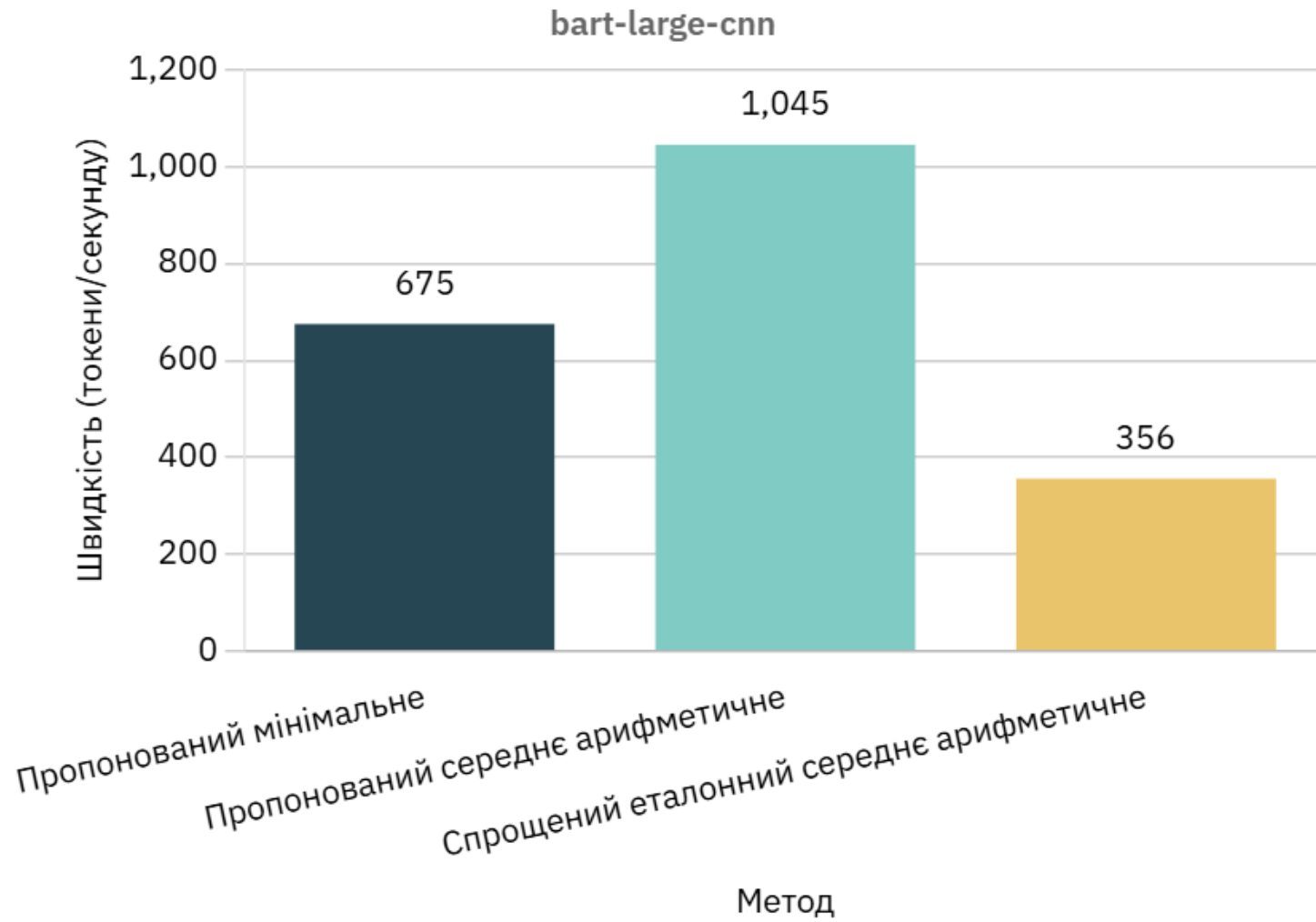
We believe in a particular order not because it is objectively true, but because believing in it enables us to cooperate effectively and forge a better society. An imagined order cannot be sustained by violence alone. The imagined order is inter-subjective. In order to change them we must simultaneously change the consciousness of billions of people.

Кінаш Дарій, КП-31мн

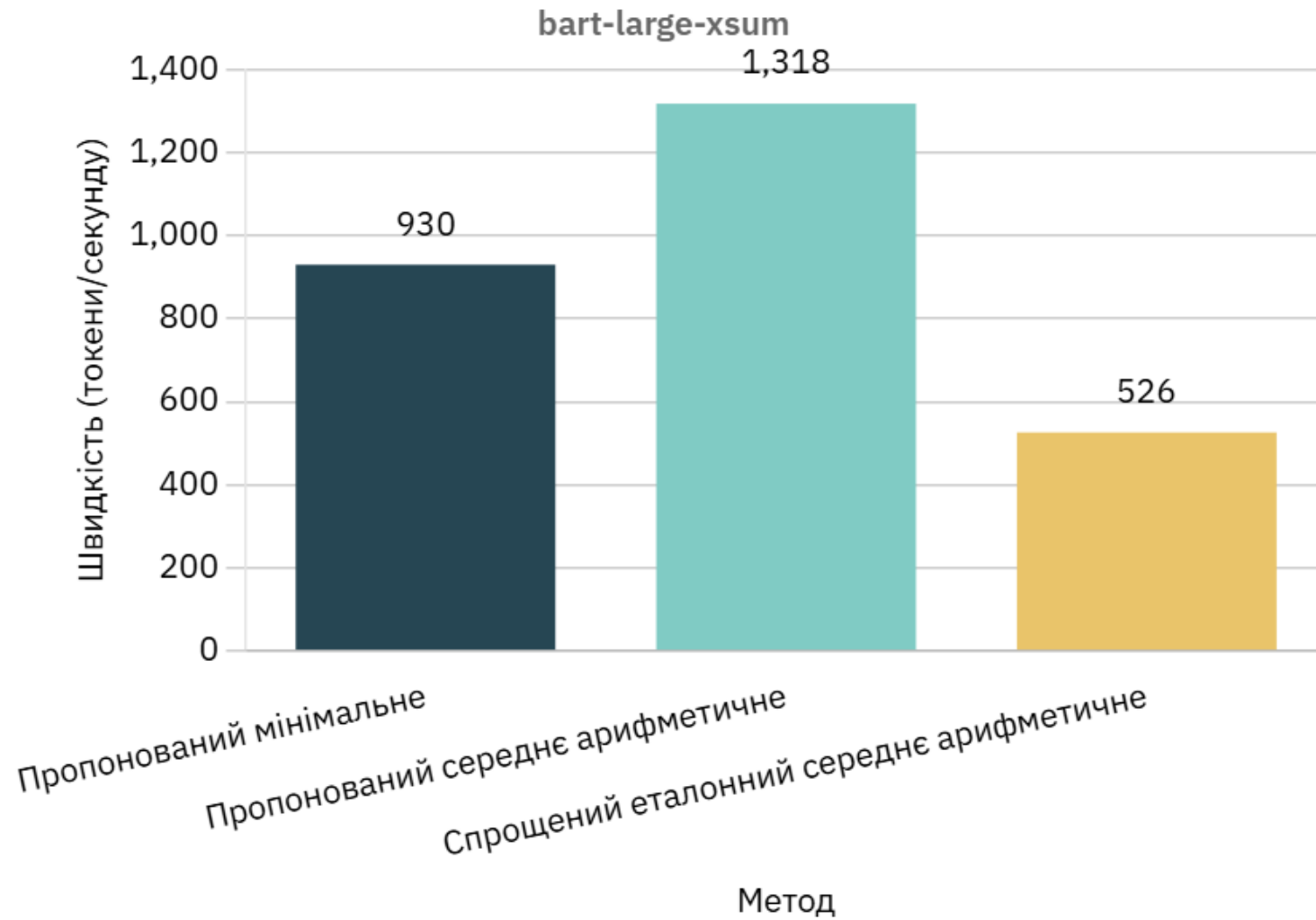
Зведене порівняння швидкості виконання для pegasus-large



Зведене порівняння швидкості виконання для bart-large-cnn



Зведене порівняння швидкості виконання для bart-large-xsum



**Додаток 2**  
**Лістинг програми**

## llm\_types.py

```
from enum import Enum

class LLMType(Enum):
    PEGASUS = 'pegasus'
    BART_CNN = 'bart_cnn'
    BART_XSUM = 'bart_xsum'
```

## llm\_interface.py

```
import abc
from typing import List

class LLMInterface(metaclass=abc.ABCMeta):

    @classmethod
    def __subclasshook__(cls, subclass):
        return (hasattr(subclass, 'get_input_limit') and
                callable(subclass.get_input_limit) and
                hasattr(subclass, 'get_chunking_cushion') and
                callable(subclass.get_chunking_cushion) and
                hasattr(subclass, 'tokenize') and
                callable(subclass.tokenize) and
                hasattr(subclass, 'summarize') and
                callable(subclass.summarize) or
                NotImplemented)

    @abc.abstractmethod
    def get_input_limit(self) -> int:
        raise NotImplementedError

    @abc.abstractmethod
    def get_chunking_cushion(self) -> int:
        raise NotImplementedError

    @abc.abstractmethod
    def tokenize(self, text: str) -> List[str]:
        raise NotImplementedError

    @abc.abstractmethod
    def summarize(self, chunks: List[str]) -> List[str]:
        raise NotImplementedError
```

## preprocessor.py

```
import re
import nltk.data
from typing import List

class Preprocessor:

    @staticmethod
    def preprocess(filename: str) -> List[str]:
        # read text from file
        raw_text = ''
        with open('../texts/' + filename + '.txt', encoding = 'utf8') as f:
            raw_text = f.read()

        # remove noise data
        noise_chars = {'*', '|', '_', '{', '}', '[', ']'}
        denoised_text = raw_text
        for char in noise_chars:
```

```

        denoised_text = denoised_text.replace(char, ' ')

no_whitespace_text = re.sub(r'\s+', ' ', denoised_text)
no_whitespace_text = no_whitespace_text.strip()
clean_text = re.sub(r"['\"]", "", no_whitespace_text)

# split text into sentences
sentence_tokenizer =
    nltk.data.load('tokenizers/punkt/english.pickle')

sentence_tokenizer._params.abbrev_types =
    set.union(
        sentence_tokenizer._params.abbrev_types,
        {'i.e', 'e.g'})
    )
all_sentences = sentence_tokenizer.tokenize(clean_text)

# remove all sentences below length threshold
sentences = []
for i in all_sentences:
    if (len(i) >= 30):
        sentences.append(i)

return sentences

```

## pegasus.py

```

from .llm_interface import LLMInterface
from typing import List
from transformers import PegasusForConditionalGeneration, PegasusTokenizer

class Pegasus(LLMInterface):

    def __init__(self):
        self.__name: str = 'google/pegasus-large'
        self.__device: str = 'cpu'
        self.__input_limit: int = 1024
        self.__chunking_cushion: int = 150
        self.__tokenizer = PegasusTokenizer.from_pretrained(self.__name)
        self.__model =
            PegasusForConditionalGeneration
                .from_pretrained(self.__name)
                .to(self.__device)

    def get_input_limit(self) -> int:
        return self.__input_limit

    def get_chunking_cushion(self) -> int:
        return self.__chunking_cushion

    def tokenize(self, text: str) -> List[str]:
        return self.__tokenizer.tokenize(text)

    def summarize(self, chunks: List[str]) -> List[str]:
        batch = self.__tokenizer(
            chunks,
            truncation=True,
            padding="longest",
            return_tensors="pt"
        ).to(self.__device)

        result = self.__model.generate(**batch)
        return self.__tokenizer.batch_decode(

```

```

        result,
        skip_special_tokens=True
    )

```

## bart.py

```

from .llm_interface import LLMInterface
from .llm_types import LLMTType
from typing import List
from transformers import BartTokenizer, BartForConditionalGeneration

class Bart(LLMInterface):

    def __init__(self, type: LLMTType):
        if (type == LLMTType.BART_CNN):
            self.__name: str = 'facebook/bart-large-cnn'
        elif (type == LLMTType.BART_XSUM):
            self.__name: str = 'facebook/bart-large-xsum'
        else:
            raise Exception('Wrong LLMTType for class Bart')

        self.__input_limit: int = 1024
        self.__chunking_cushion: int = 150
        self.__tokenizer = BartTokenizer.from_pretrained(self.__name)
        self.__model =
            BartForConditionalGeneration.from_pretrained(self.__name)

    def get_input_limit(self) -> int:
        return self.__input_limit

    def get_chunking_cushion(self) -> int:
        return self.__chunking_cushion

    def tokenize(self, text: str) -> List[str]:
        return self.__tokenizer.tokenize(text)

    def summarize(self, chunks: List[str]) -> List[str]:
        batch = self.__tokenizer(
            chunks,
            truncation=True,
            padding="longest",
            return_tensors="pt",
            max_length=self.__input_limit
        )
        result = self.__model.generate(
            batch["input_ids"],
            num_beams=2,
            min_length=0,
            max_length=150
        )
        return self.__tokenizer.batch_decode(
            result,
            skip_special_tokens=True,
            clean_up_tokenization_spaces=False
        )

```

## app.py

```

import tensorflow as tf
import tensorflow_hub as hub
import hdbscan
import math
import argparse

```

```

from datetime import datetime
from llm_impl.llm_types import LLMType
from llm_impl.pegasus import Pegasus
from llm_impl.bart import Bart
from preprocessor import Preprocessor

sentence_embedding_block_max_size = 1000

parser = argparse.ArgumentParser()
parser.add_argument('--llm', type = LLMType, choices = list(LLMType),
                    required = True)
parser.add_argument('--file', type = str, required = True)
args = parser.parse_args()

llm = None
if (args.llm == LLMType.PEGASUS):
    llm = Pegasus()
else:
    llm = Bart(args.llm)

script_start_ts = datetime.now()

sentences = Preprocessor.preprocess(args.file)

# to count the speed of summarization
total_tokens = 0
for i in sentences:
    total_tokens = total_tokens + len(llm.tokenize(i))

embedding_start_ts = datetime.now()

# generate vector representations (embeddings) for sentences
# https://www.kaggle.com/models/google/universal-sentence-
encoder/TensorFlow2/large/2
embedding_model = hub.load('../models/use-large-tf2')
embeddings = None
for i in range(math.ceil(len(sentences) /
sentence_embedding_block_max_size)):
    temp_embeddings = embedding_model(
        sentences[(i * sentence_embedding_block_max_size):((i + 1) *
sentence_embedding_block_max_size)]
    )
    if (i == 0):
        embeddings = temp_embeddings
    else:
        embeddings = tf.concat([embeddings, temp_embeddings], axis = 0)

# (debug - optional)
# check if embeddings are actually normalized
# magnitudes of some may be slightly bigger or smaller than 1
# due to floating point rounding errors
'''
import numpy as np
for embedding in embeddings:
    vector = np.array(embedding)
    magnitude = np.linalg.norm(vector)
    if (magnitude > 1.00001 or magnitude < 0.99999):
        print('Embedding is not normalized: ')
        print(magnitude)
'''

```

```

clustering_start_ts = datetime.now()

# cluster the embeddings
clusterer = hdbscan.HDBSCAN(
    min_cluster_size = 10,
    min_samples = 2,
    cluster_selection_method = 'leaf',
    allow_single_cluster=True
).fit(embeddings)

cluster_number = max(clusterer.labels_) + 1

if (cluster_number == 0):
    print('EXIT - extracted 0 clusters')
    exit()

# append each sentence that was clustered
# into corresponding cluster sentence array
# in order the sentences appear in the original text
total_clustered_sentences = 0
clusters = []

for i in range(cluster_number):
    clusters.append({'sentences': [], 'token_count': 0, 'index_sum': 0,
                    'summary': '', 'chunk_count': 0, 'llm_time': ''})

for i in range(len(clusterer.labels_)):
    label = clusterer.labels_[i]
    if (label >= 0):    ### filter by probability here if needed
        token_count = len(llm.tokenize(sentences[i]))
        clusters[label]['sentences'].append(
            {'sentence': sentences[i], 'probability':
clusterer.proBABILITIES_[i], 'index': i, 'token_count': token_count}
        )
        clusters[label]['token_count'] = clusters[label]['token_count'] +
token_count
        clusters[label]['index_sum'] = clusters[label]['index_sum'] + i
        total_clustered_sentences = total_clustered_sentences + 1

llm_start_ts = datetime.now()

for cluster in clusters:
    token_count = cluster['token_count']
    text = ''
    chunks = []
    if (token_count <= llm.get_input_limit()):
        for data in cluster['sentences']:
            text = text + data['sentence'] + ' '
            chunks.append(text.strip())
    else:
        # chunk count will not necessarily be equal to computed value
        chunk_count = math.ceil(token_count / (llm.get_input_limit() -
llm.get_chunking_cushion()))
        chunk_size = math.floor(token_count / chunk_count)
        sent_cnt = len(cluster['sentences'])
        curr_tokens = 0
        for i in range(sent_cnt):
            data = cluster['sentences'][i]
            if (curr_tokens > chunk_size or curr_tokens +
data['token_count'] > llm.get_input_limit()):
                chunks.append(text.strip())
                curr_tokens = 0

```

```

        text = ''
        text = text + data['sentence'] + ' '
        curr_tokens = curr_tokens + data['token_count']
        if (i == sent_cnt - 1):
            chunks.append(text.strip())

cluster['chunk_count'] = len(chunks)

start_ts = datetime.now()
summaries = llm.summarize(chunks)
cluster['llm_time'] = str(datetime.now() - start_ts)

for sum in summaries:
    cluster['summary'] = cluster['summary'] + sum + '\n'

sorting_start_ts = datetime.now()

# sort clusters by sentence order in original text
clusters.sort(key = lambda obj: obj['index_sum'] / len(obj['sentences']))

script_end_ts = datetime.now()

file_name = args.file + '_' + args.llm.value + '_' +
str(script_start_ts.replace(microsecond=0)).replace(' ', '_').replace(':',
'-') + '.txt'

# write summarization results into file
with open('../output/res/' + file_name, mode = 'w', encoding = 'utf8') as
f:

    f.write('\nFile:                ' + args.file +
            '\nLLM:                  ' + args.llm.value +
            '\nToken count:                 ' + str(total_tokens) +
            '\nSpeed (tokens/s):            ' + str(total_tokens / (script_end_ts
- script_start_ts).total_seconds()) +
            '\nCluster count:               ' + str(cluster_number) +
            '\nTotal sentences:               ' + str(len(sentences)) +
            '\nClustered sentences:          ' + str(total_clustered_sentences) +
            '\nClustered ( % ):              ' +
str(math.ceil(total_clustered_sentences * 100 / len(sentences))) +
            '\nPreprocessing time:          ' + str(embedding_start_ts -
script_start_ts) +
            '\nEmbedding time:              ' + str(clustering_start_ts -
embedding_start_ts) +
            '\nClustering time:            ' + str(llm_start_ts -
clustering_start_ts) +
            '\nLLM time:                  ' + str(sorting_start_ts -
llm_start_ts) +
            '\nSorting time:              ' + str(script_end_ts -
sorting_start_ts) +
            '\nTotal time:                ' + str(script_end_ts -
script_start_ts))

    f.write('\n\n-----SUMMARY START-----\n\n\n')

    for cluster in clusters:
        f.write(cluster['summary'] + '\n')

    f.write('\n\n\n-----SUMMARY END-----\n\n\n')

f.write('\n\nprobability ; tokens ; original index\n\n')
for i in range(len(clusters)):

```

```

        sentence_count = len(clusters[i]['sentences'])
        f.write('Cluster ' + str(i + 1) + ' ( ' + str(sentence_count) + '
sentences; ' +
str(clusters[i]['token_count']) + ' tokens; ' +
str(math.floor(clusters[i]['index_sum'] / sentence_count)) + ' position; '
+
str(clusters[i]['chunk_count']) + ' chunks; ' +
clusters[i]['llm_time'] +
' exec time)\n')

        f.write('\n' + clusters[i]['summary'] + '\n')

        for data in clusters[i]['sentences']:
            f.write('\t(' + str(round(data['probability'], 2)).rjust(4, '
') + ';' +
str(data['token_count']).rjust(4, ' ') + ';' +
str(data['index']).rjust(6, ' ') + ')\t' +
data['sentence'] + '\n')

        f.write('\n\n')

```

**Додаток 3**  
**Копія презентації**



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ  
СІКОРСЬКОГО”



ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

**МЕТОД ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ  
УЗАГАЛЬНЕННЯ ВЕЛИКОРОЗМІРНИХ НАУКОВО-  
ПОПУЛЯРНИХ ТЕКСТІВ**

Доповідач: Кінаш Дарій Олегович

Науковий керівник: к.т.н., доц., доцент кафедри ПЗКС  
Заболотня Тетяна Миколаївна

Київ – 2025

## АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ



- Короткий виклад змісту тексту дозволяє отримати ключові ідеї, теми та суть тексту у спрощеному вигляді без перегляду всього тексту, що дозволяє уникнути втрати часу та інформаційного перенасичення.
- Наявні на даний момент методи узагальнення великорозмірних текстів, що створюють зв'язне та послідовне скорочення, яке охоплює ключові теми вхідного тексту, мають низьку швидкість роботи та не пристосовані до виконання на пристроях користувачів.



## ПОСТАНОВКА ЗАДАЧІ

**Об'єкт дослідження:** процес автоматизованого аналізу природномовних даних.

**Предмет дослідження:** методи, способи, програмні засоби узагальнення природномовного тексту.

**Мета дослідження:** підвищення швидкості процесу узагальнення великорозмірних науково-популярних текстів при збереженні таких якісних характеристик результату узагальнення, як зв'язність, послідовність та охоплення ключових тем.



## ОКРЕМІ ЗАВДАННЯ

- Дослідити та проаналізувати наявні методи узагальнення великорозмірного тексту, зосередившись на швидкості їхньої роботи та якісних характеристиках їхнього результату узагальнення.
- Обрати еталонний метод для порівняння ефективності, що забезпечує зв'язність, послідовність та охоплення ключових тем вхідного тексту.
- Дослідити характерні риси науково-популярних текстів.
- Розробити метод для узагальнення великорозмірних науково-популярних текстів з покращеною швидкістю роботи порівняно з обраним еталонним методом.
- Обрати програмні інструменти для реалізації запропонованого методу.
- Створити програмне забезпечення, яке реалізує метод.
- Провести експерименти для оцінювання ефективності розробленого методу.



## ТЕРМІНОЛОГІЯ

- Екстрактне узагальнення (реферування) – стиснення тексту до мінімального розміру шляхом обрання найбільш значущих речень. Може реалізовуватись як алгоритмічними методами, так і моделями машинного навчання.
- Абстрактне узагальнення (анотування) – стиснення тексту до мінімального розміру шляхом перефразування найбільш контекстно-значущих частин тексту. Реалізується моделями машинного навчання.
- Зв'язність тексту – це властивість, яка характеризує структурну цілісність всередині речень та між окремими реченнями і забезпечує їхню взаємодію та плавний перехід.
- Послідовність тексту – це властивість, яка характеризує логічну та змістову цілісність і чітку організацію інформації між реченнями, тобто те, як кожне наступне речення або частина тексту логічно впливає з попереднього і підтримує єдину тему або основну ідею блоку речень.

# АНАЛІЗ НАЯВНИХ МЕТОДІВ



	Висока швидкість обробки	Немає обмежень на обсяг вхідного тексту	Зв'язний результат	Послідовний результат	Повне охоплення тематичних блоків тексту
Екстрактне узагальнення за допомогою базових методів	Так	Так	Ні	Ні	Ні
Абстрактне узагальнення урізаного тексту	Так	Так	Так	Ні	Ні
Комбінований метод з застосуванням екстрактного та абстрактного узагальнення	Так	Так	Так	Ні	Ні
Екстрактне узагальнення з застосуванням кластеризації	Так	Так	Ні	Ні	Так
Багатокрокове абстрактне узагальнення з застосуванням поділу тексту на фрагменти	Ні	Так	Так	Так	Так
Абстрактне узагальнення з застосуванням моделі лонгформера	Так	Ні	Так	Ні	Ні

# ПРОПОНОВАНИЙ МЕТОД



Пропонований комбінований метод передбачає два послідовні етапи:

1. Застосування екстрактного узагальнення шляхом кластеризації векторних відображень речень та обрання найбільш репрезентативних речень з кожного кластеру. Як наслідок розмір тексту зменшується в середньому в 10 разів, що забезпечує вищу швидкість роботи.
2. Застосування абстрактного узагальнення з використанням фрагментації до кожного кластера речень та об'єднання отриманих скорочень в кінцевий результат.

# ПРОПОНОВАНИЙ МЕТОД



Основні етапи пропонованого методу:

1. Попереднє оброблення тексту.
2. Розбиття тексту на речення.
3. Обчислення векторних відображень речень.
4. Кластеризація векторних відображень.
5. Обрання найбільш репрезентативних речень з кожного кластера.
6. Виконання абстрактного узагальнення з застосуванням фрагментації для кожної групи речень.
7. Сортування результатів узагальнення.
8. Конкатенація в кінцевий результат узагальнення.



# ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ



Обрана метрика для оцінювання ефективності – швидкість роботи.

$$V = N/t$$

- $N$  – кількість tokenів у вхідному тексті;
- $t$  – загальний час виконання методу.

Порівняння пропонованого методу буде проводитись зі методом багатокрокового абстрактного узагальнення з застосуванням фрагментації визначеної контекстом.

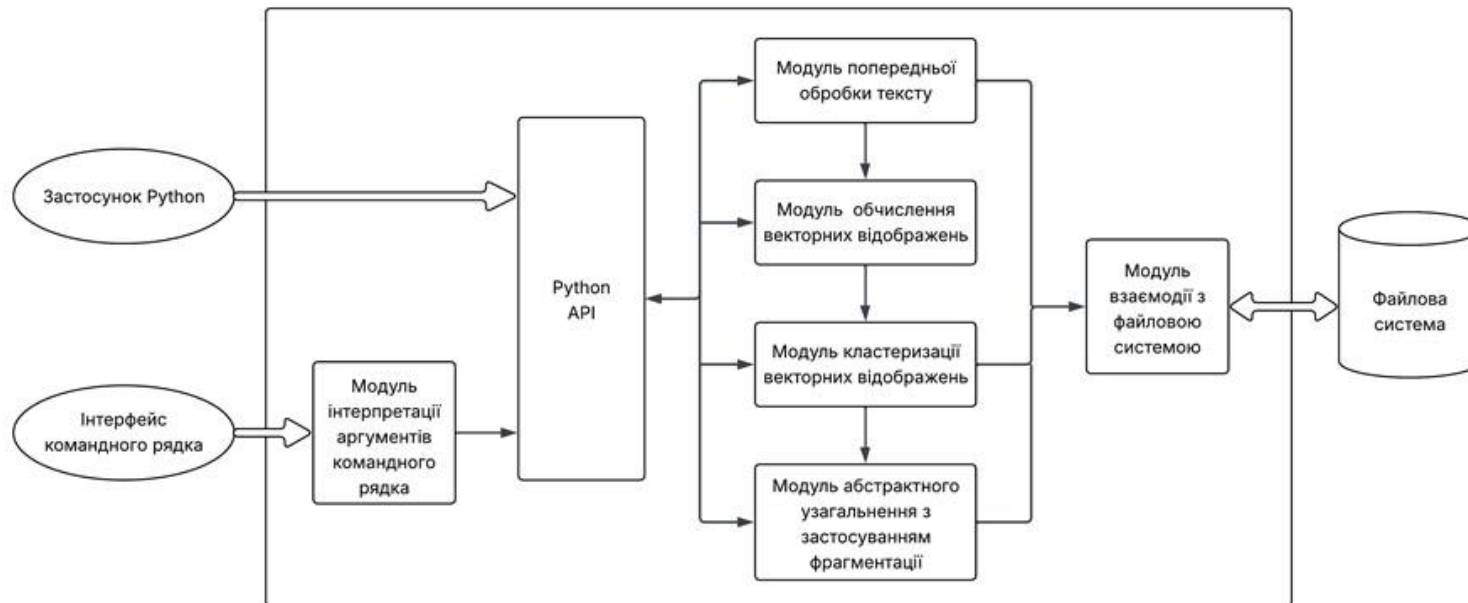
Оцінка якості узагальнення вручну замість автоматизованої перевірки метриками на кшталт ROUGE, BLEU, METEOR та BERTScore через обмеження метрик та відсутність датасетів з еталонними узагальненнями для науково-популярних текстів.

# ІНСТРУМЕНТИ РЕАЛІЗАЦІЇ

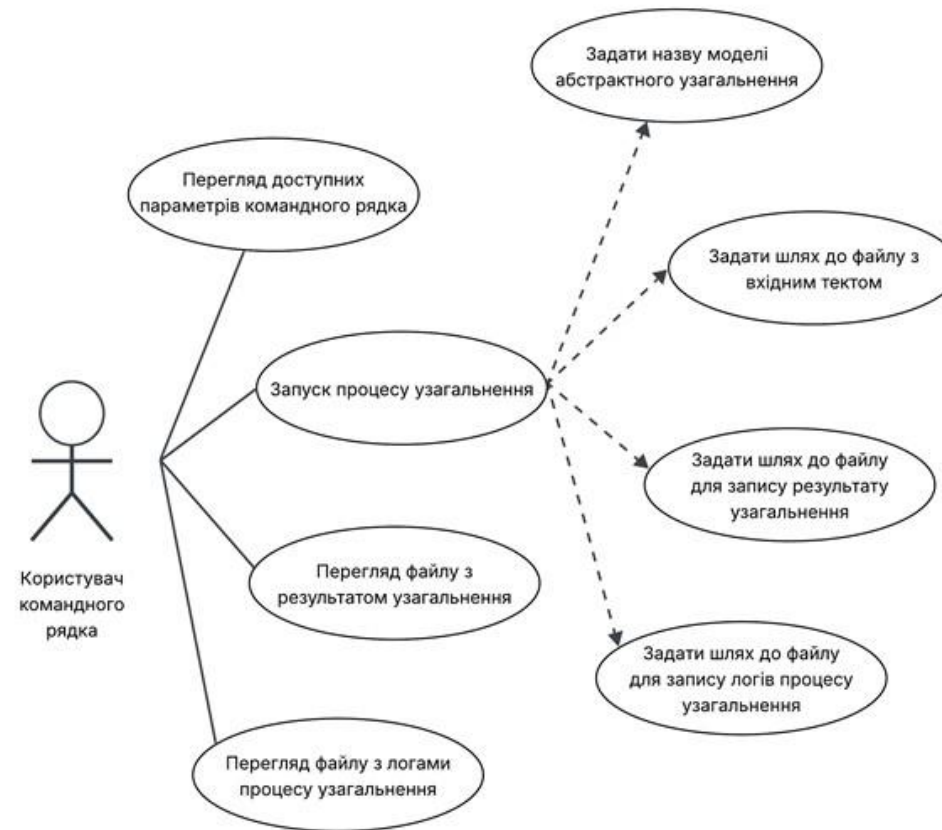


HUGGING FACE

# ДІАГРАМА МОДУЛІВ РОЗРОБЛЕНОГО ПЗ



# СЦЕНАРІЙ ВИКОРИСТАННЯ РОЗРОБЛЕНОГО ПЗ



## ПРОВЕДЕННЯ ОЦІНЮВАННЯ



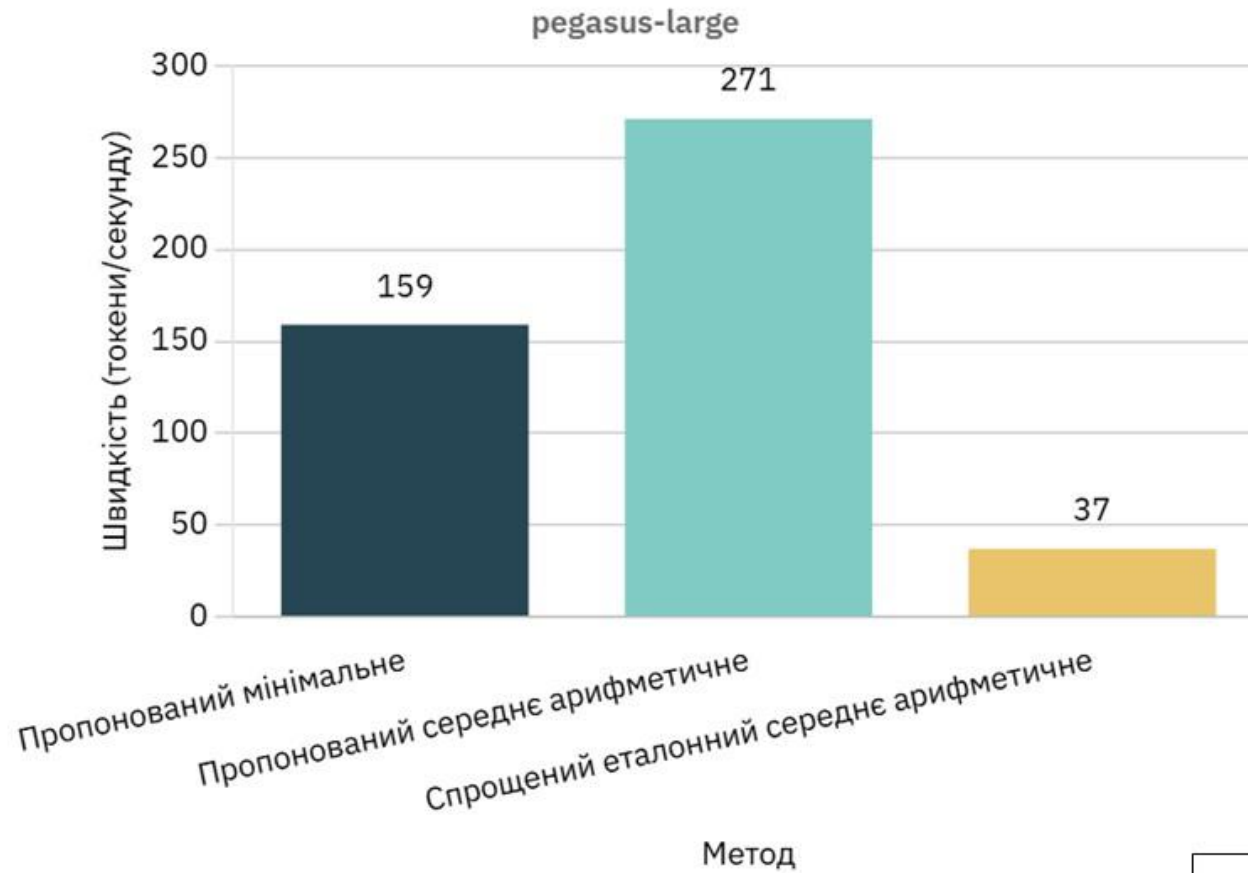
Для більш об'єктивного порівняння швидкості роботи оцінювання було проведене з застосуванням моделей *pegasus-large*, *bart-large-cnn*, *bart-large-xsum*.

Оцінювання швидкості було проведене на 50 текстах, оцінювання якості узагальнення було проведено на наступних 5 текстах:

1. *“Sapiens: A Brief History of Humankind”* авторства Ювала Ноя Харарі.
2. *“A Brief History of Time”* авторства Стівена Гокінга.
3. *“The Elegant Universe”* авторства Браяна Гріна.
4. *“Thinking, Fast and Slow”* авторства Деніела Канемана.
5. *“The Uninhabitable Earth: Life After Warming”* авторства Девіда Воллес-Веллса.

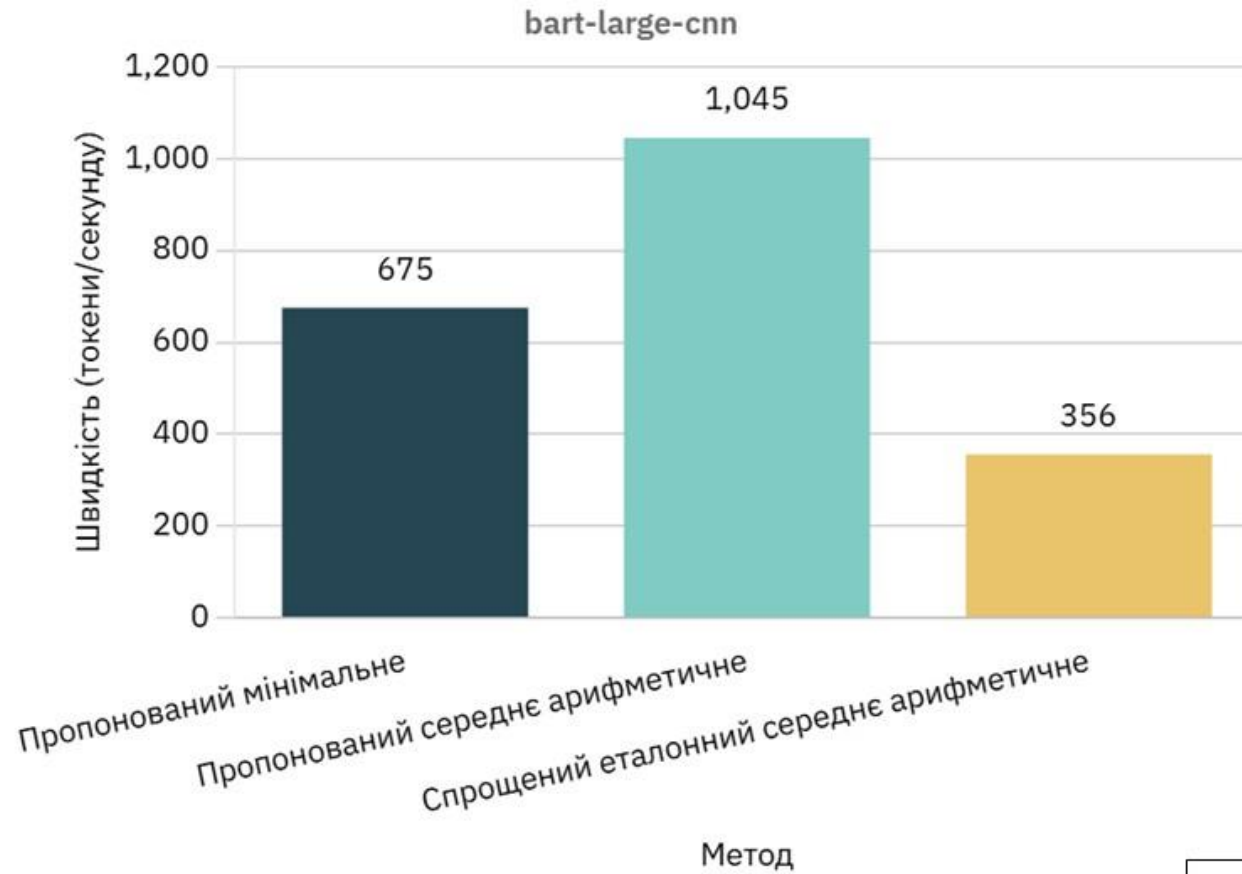


# РЕЗУЛЬТАТИ ОЦІНЮВАННЯ

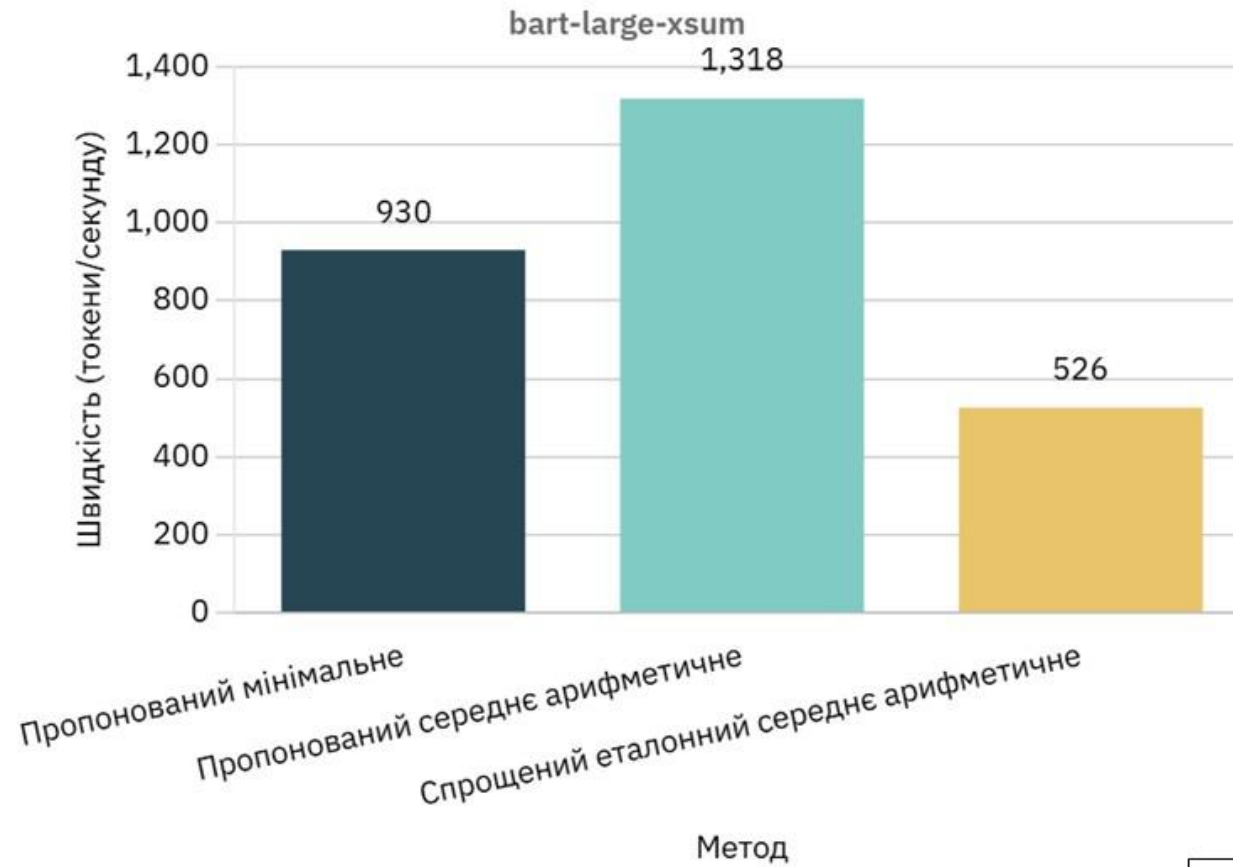




# РЕЗУЛЬТАТИ ОЦІНЮВАННЯ



# РЕЗУЛЬТАТИ ОЦІНЮВАННЯ



## ПОРІВНЯННЯ ЯКОСТІ РЕЗУЛЬТАТІВ УЗАГАЛЬНЕННЯ



- Зв'язність тексту співставна для пропонованого та еталонного методу.
- При виконанні одної ітерації еталонного методу у його результаті краще охоплення тем, присутнє включення більш обширних та абстрактних тем та краще відображається логічна послідовність, однак завелика довжина.
- При виконанні двох чи більше ітерацій еталонного методу у його результаті гірше охоплення тем, погано відображається логічна послідовність та замала довжина.
- Пропонований метод має незначне зниження якості узагальнення через відсутність певних тематичних блоків, однак має прийнятну довжину.

# ПРИКЛАД РЕЗУЛЬТАТУ УЗАГАЛЬНЕННЯ



Biologists label organisms with a two-part Latin name, genus followed by species. Animals are said to belong to the same species if they tend to mate with each other, giving birth to fertile offspring. Horses and donkeys have a recent common ancestor and share many physical traits.

The legend of Peugeot affords us a good example. In what sense can we say that Peugeot SA (the company's official name) exists? There are many Peugeots, but these are obviously not the company. Even if every Peugeot in the world were simultaneously junked and sold for scrap metal, Peugeot SA would not disappear.

Chimps can't win an argument with a Homo sapiens, but the ape can rip the man apart like a rag doll. The alpha male usually wins his position not because he is physically stronger, but because he leads a large and stable coalition.

Humans first evolved in East Africa about 2.5 million years ago from an earlier genus of apes called Australopithecus. Humans in Europe and western Asia evolved into Homo neanderthalensis. Homo erectus, 'Upright Man', survived there for close to 2 million years. According to this view, Sapiens replaced all the previous human populations without merging with them. If that is the case, the lineages of all contemporary humans can be traced back, exclusively, to East Africa, 70,000 years ago. But if the Interbreeding Theory is right, there might well be genetic differences between Africans, Europeans and Asians that go back hundreds of thousands of years. Homo erectus did not undergo further genetic alterations, its stone tools remained roughly the same -- for close to 2 million years. In a one-on-one brawl, a Neanderthal would probably have beaten a Sapiens. But in a conflict of hundreds, Neanderthals wouldn't stand a chance. Homo sapiens evolved to think of people as divided into us and them. Evolution has made Homo sapiens, like other social mammals, a xenophobic creature. The Nazis explained that Homo sapiens itself appeared when one 'superior' population of ancient humans evolved.

If a superior man should blind the eye of another superior 197.5 If a superior man strikes a woman of superior class and thereby causes her to miscarry her fetus, he shall weigh and deliver ten shekels of silver for her fetus. If that woman should die, they shall kill his daughter.

We believe in a particular order not because it is objectively true, but because believing in it enables us to cooperate effectively and forge a better society. An imagined order cannot be sustained by violence alone. The imagined order is inter-subjective. In order to change them we must simultaneously change the consciousness of billions of people.

## НАУКОВА НОВИЗНА



Вперше запропоновано метод для узагальнення великорозмірних науково-популярних текстів, що створює зв'язний та послідовний результат, який охоплює ключові теми вхідного тексту, та має як мінімум в 1.8 разів більшу швидкість роботи, ніж багатокроковий метод абстрактного узагальнення з застосуванням фрагментації визначеної контекстом, завдяки суттєвому зменшенню розміру вхідного тексту за рахунок застосування кластеризації та, як наслідок, зменшення кількості викликів до моделі абстрактного узагальнення.



## ПРАКТИЧНА ЦІННІСТЬ

Розроблене програмне забезпечення має наступні способи застосування:

- використання на особистих користувацьких пристроях як окремої утиліти;
- використання в межах аналітичних систем;
- використання на ресурсах для збереження та розповсюдження текстових даних.

Запропоноване рішення забезпечує вищу швидкість виконання узагальнення ніж інші рішення, що мають співставні якісні характеристики результату узагальнення.

# АПРОБАЦІЯ



Тези доповіді “Комбінований метод для узагальнення великорозмірних текстів” на конференції «Прикладна математика та комп’ютинг. ПМК-2024».

## ПОДАЛЬША РОБОТА



- Повноцінне тренування використаних моделей.
- Забезпечення можливості виконання на графічному процесорі.
- Зменшення вимірності векторних відображень речень для покращення кластеризації.
- Перевірка інших видів великорозмірних текстів на сумісність з запропонованим методом.



## ВИСНОВКИ

1. Проаналізовано наявні методи узагальнення великорозмірних текстів.
2. Розроблено комбінований метод для узагальнення великорозмірних науково-популярних текстів, який має підвищену швидкість роботи порівняно з еталонним методом, при цьому забезпечуючи співставні якісні характеристики результату узагальнення.
3. Розроблено програмну реалізацію запропонованого методу у вигляді консольного додатку на мові Python з використанням сучасних лінгвістичних програмних пакетів та бібліотек для роботи з текстовими даними.
4. Запропонований метод дозволяє підвищити швидкість роботи в 1.8 разів порівняно з еталонним методом.
5. Визначено напрямки подальшого дослідження.



*Дякую за увагу!*