Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота магістра

на тему: «Порівняльний аналіз ефективності сучасних методів кластеризації текстових даних»

Виконав: студент групи К23-1м

Спеціальність 122 Комп'ютерні науки

Клименко І.В.
(прізвище та ініціали)

Керівник д.е.н., проф. Корнєєв М.В.
(науковий ступінь, вчене звання, прізвище та ініціали)

Рецензент Дніпровський державний технічний університет
(місце роботи)

в.о. завідувача кафедри програмного забезпечення систем
(посада)

к.т.н., доц. Жульковський О.О.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2025

АНОТАЦІЯ

Клименко І.В. Порівняльний аналіз ефективності сучасних методів кластеризації текстових даних.

Дипломна робота на здобуття освітнього ступеня магістр за спеціальністю 122 «Комп'ютерні науки» – Університет митної справи та фінансів, Дніпро, 2025.

Магістерська робота присвячена дослідженню сучасних алгоритмів кластеризації для групування текстових даних. У роботі розглянуто основні підходи до кластеризації текстів, включаючи традиційні методи, такі як K-середні та ієрархічна кластеризація, а також сучасні алгоритми, які базуються на глибокому навчанні та вбудовуванні текстів. Основна увага приділяється аналізу ефективності цих підходів для вирішення практичних задач, таких як автоматична категоризація документів, тематичне моделювання, аналіз настроїв та побудова рекомендаційних систем.

У роботі виконано розробку програмного забезпечення для реалізації кластеризаційних алгоритмів та проведення експериментального порівняння їхньої ефективності на реальних текстових наборах. Проведено оцінку якості кластеризації з використанням відповідних метрик, таких як силуетні бали та внутрішньокластерні відстані, а також візуалізацію результатів для глибшого розуміння структури даних. Результати дослідження демонструють, що правильний вибір алгоритму кластеризації, метрики подібності та методу векторизації тексту є ключовими аспектами для досягнення високої точності групування текстових даних.

Ключові слова: кластеризація текстових даних, K-середні, DBSCAN, тематичне моделювання, аналіз настроїв, алгоритми глибокого навчання, обробка природної мови.

ABSTRACT

Klymenko I.V. Comparative analysis of the effectiveness of modern methods of text data clustering.

Diploma thesis for obtaining a master's degree in specialty 122 «Computer Science» – University of Customs and Finance, Dnipro, 2025.

The master's thesis is devoted to the study of modern clustering algorithms for grouping text data. The work reviews the main approaches to text clustering, including traditional methods such as K-means and hierarchical clustering, as well as modern algorithms based on deep learning and text embedding. The main focus is on analyzing the effectiveness of these approaches for solving practical problems, such as automatic document categorization, topic modeling, sentiment analysis, and building recommender systems.

In this work, we develop software for implementing clustering algorithms and conduct an experimental comparison of their effectiveness on real text sets. The quality of clustering is evaluated using relevant metrics such as silhouette scores and intra-cluster distances, and the results are visualized for a deeper understanding of the data structure. The results of the study demonstrate that the correct choice of clustering algorithm, similarity metric and text vectorization method are key aspects for achieving high accuracy of text data grouping.

Keywords: text data clustering, K-means, DBSCAN, topic modeling, sentiment analysis, deep learning algorithms, natural language processing.

ЗМІСТ

ВСТУП

У сучасному світі, де обсяг інформації зростає експоненційно, питання ефективної обробки та аналізу текстових даних стає дедалі важливішим. Великі масиви тексту, такі як новини, повідомлення в соціальних мережах, наукові статті чи клієнтські відгуки, є цінним джерелом знань, проте їхнє опрацювання вимагає застосування сучасних методів обробки даних. Алгоритми кластеризації, як один із ключових інструментів машинного навчання, відіграють важливу роль у вирішенні цієї проблеми, дозволяючи автоматично групувати текстові дані за схожістю [1].

Актуальність дослідження обумовлена необхідністю розвитку та впровадження ефективних алгоритмів кластеризації, які здатні враховувати особливості текстових даних, такі як високий ступінь неоднорідності, багатовимірність та залежність від контексту. Успішна реалізація таких підходів має важливе значення для ряду практичних завдань: аналізу настроїв споживачів, автоматичної категоризації документів, створення рекомендаційних систем, а також для автоматизації рутинних процесів у сферах освіти, медицини, бізнесу та інших галузях. Водночас сучасні методи кластеризації постійно вдосконалюються, що вимагає глибокого вивчення їхніх переваг, недоліків та сфер застосування.

Метою даної кваліфікаційної роботи є аналіз і порівняння сучасних алгоритмів кластеризації для групування текстових даних, а також визначення найбільш ефективних підходів у залежності від специфіки поставленого завдання.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– проаналізувати основні типи алгоритмів кластеризації, їхні принципи роботи та застосування у контексті текстових даних;

– дослідити переваги та обмеження сучасних підходів до кластеризації тексту, включаючи методи на основі машинного навчання та глибокого навчання;

– розробити та реалізувати експериментальну методику порівняння ефективності різних алгоритмів кластеризації;

– провести експериментальну оцінку алгоритмів на основі реальних текстових корпусів і проаналізувати отримані результати;

– надати рекомендації щодо вибору оптимальних алгоритмів для різних прикладних завдань.

Об’єктом дослідження є процеси кластеризації текстових даних у рамках задач аналізу та обробки природної мови.

Предметом дослідження є сучасні алгоритми кластеризації, їхні принципи роботи та ефективність у вирішенні завдань групування текстових даних.

У процесі виконання роботи використано наступні методи дослідження:

– теоретичний аналіз літератури для вивчення концептуальних основ і сучасних підходів до кластеризації;

– методи обробки природної мови (NLP) для попередньої підготовки текстових даних;

– експериментальне моделювання для порівняння алгоритмів;

– статистичний аналіз результатів для оцінки ефективності алгоритмів.

Практична значимість роботи полягає в тому, що її результати можуть бути використані для побудови ефективних систем кластеризації текстових даних у різних галузях, включаючи інформаційні технології, маркетинг, наукові дослідження тощо. Це дозволить підвищити точність і швидкість аналізу тексту, зменшити витрати на обробку даних і сприяти автоматизації багатьох рутинних завдань.

Наукова новизна дослідження полягає у комплексному аналізі сучасних алгоритмів кластеризації текстових даних з акцентом на новітні підходи, такі

як методи глибокого навчання. Особлива увага приділена порівнянню ефективності цих алгоритмів на реальних текстових наборах, що дозволило визначити їх сильні та слабкі сторони у конкретних сценаріях застосування.

Кваліфікаційна робота спрямована на вирішення актуальної науково-прикладної проблеми, яка має важливе значення для подальшого розвитку технологій обробки текстової інформації.

Структура кваліфікаційної роботи. Кваліфікаційна робота складається з трьох розділів, об’єм роботи – 92 сторінки, робота містить 19 рисунків. Список використаних джерел має 16 посилань.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Поняття кластеризації та її застосування

K-means – базова концепція в аналізі даних і машинному навчанні, згідно з якою набір об’єктів поділяється на кластери відповідно до схожості, так що об’єкти в одному кластері більше схожі один на одного, ніж на об’єкти в інших кластерах [1]. Це тип неконтрольованого навчання, при якому не потрібні позначені дані, а намагаються знайти приховані шаблони або структури в даних. Основна ідея кластеризації полягає в мінімізації внутрішньокластерної дисперсії та максимізації міжкластерної дисперсії – по суті, для того, щоб точки даних у кожній групі були якнайближчими, а групи відрізнялися одна від одної.

Застосування кластеризації охоплює широкий спектр сфер, від біології та медицини до маркетингу та соціальних наук. Наприклад, у біоінформатиці кластеризація використовується для групування генів із подібними моделями експресії, допомагаючи у відкритті біологічних шляхів і механізмів захворювання. У маркетингу кластеризація здійснюється для сегментації клієнтів за їхньою поведінкою, щоб компанії могли точніше розробляти свої продукти та рекламні стратегії відповідно до потреб і вподобань певних споживачів. Ще одним ключовим застосуванням кластеризації є сегментація зображення, де вона використовується для поділу зображення на значущі області, що може дозволити виконувати завдання більш високого рівня, такі як розпізнавання об’єктів або інтерпретація сцени.

На рисунку 1.1 наведено приклад кластеризації даних. Порівняння різноманітних алгоритмів кластеризації наведено на рисунку 1.2.

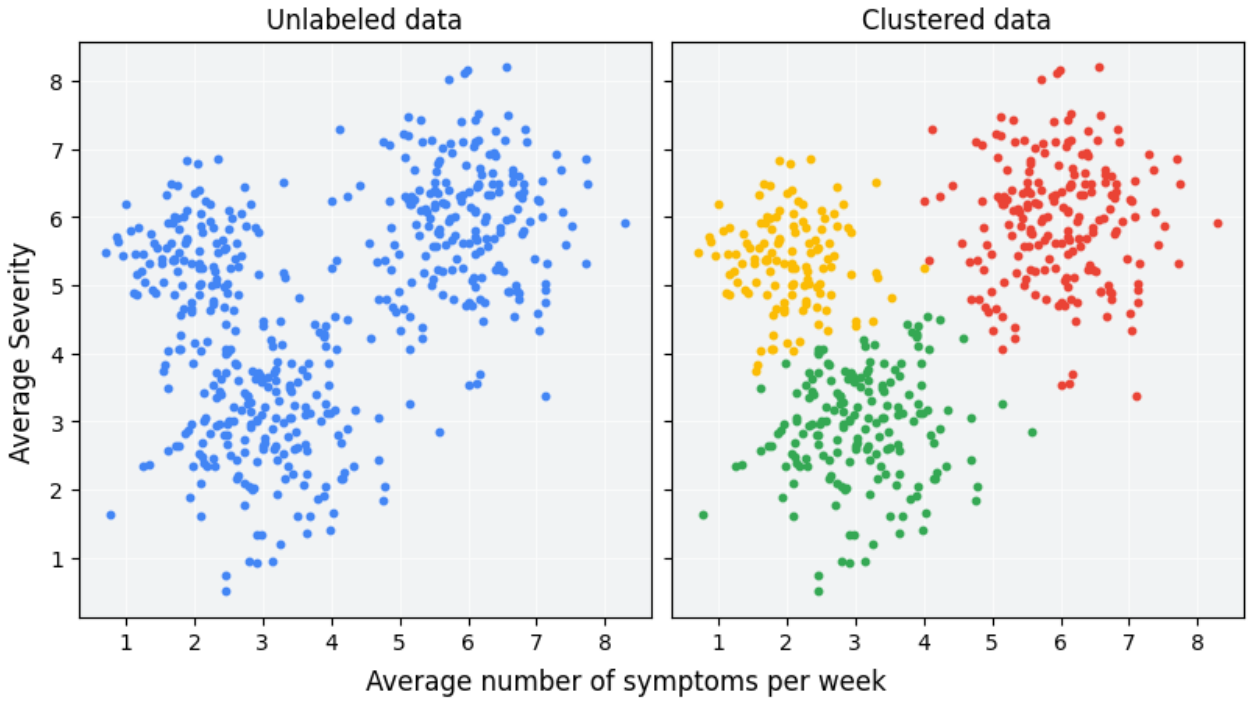

Рисунок 1.1 – Приклад кластеризації

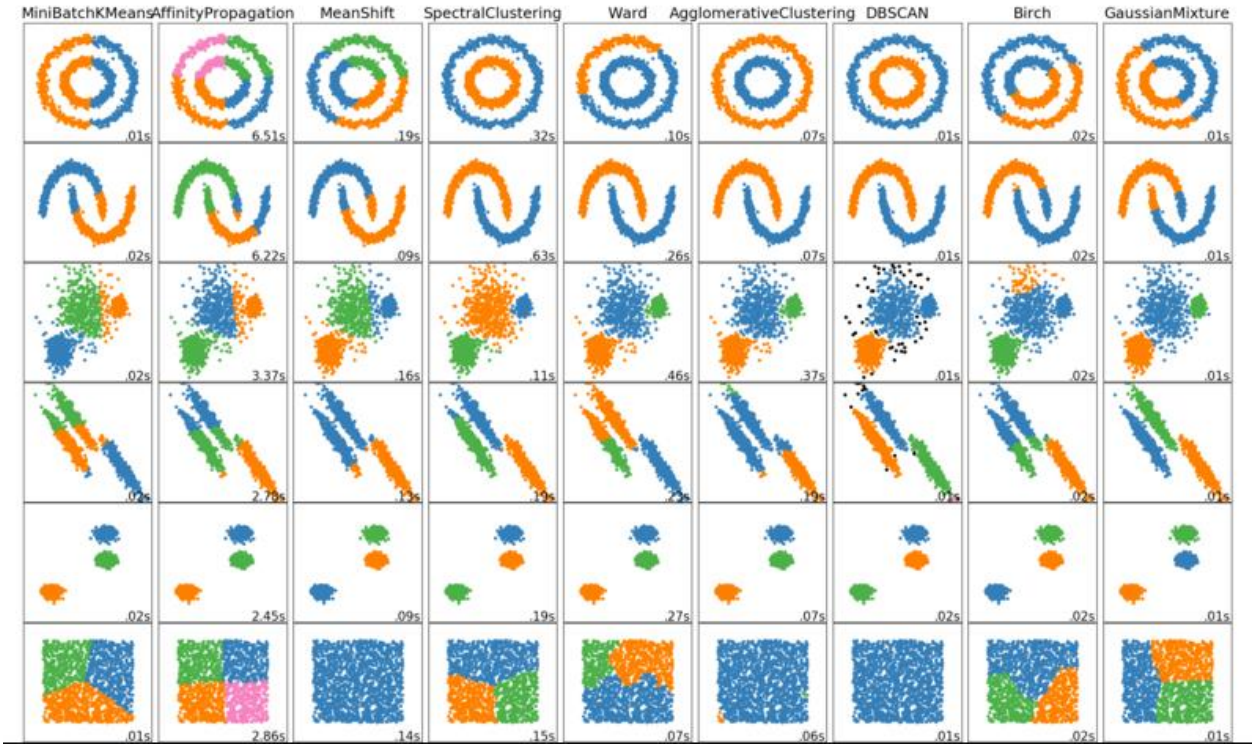

Рисунок 1.2 – Порівняння алгоритмів кластеризації

Кластеризація зазвичай включає ряд кроків: збір даних і попередню обробку. Цей крок дуже важливий, оскільки багато необроблених даних

потрібно очистити та нормалізувати для порівнянності функцій. Потім слід вибрати відповідний алгоритм кластеризації. До популярних належать такі методи, як k-середні, ієрархічна кластеризація та DBSCAN. Більшість алгоритмів вибираються на основі характеру даних, кількості очікуваних кластерів і бажаного результату [1, 2]. Наприклад, кластеризація K-середніх вимагає попередньо визначеної кількості кластерів і найкраще працює у випадках, коли кластери мають сферичну форму та мають приблизно однакові розміри. З іншого боку, ієрархічна кластеризація створює деревоподібну структуру і є більш доцільною, коли кількість кластерів невідома заздалегідь, що дозволяє більш гнучкий підхід до формування груп. DBSCAN, або просторова кластеризація додатків із шумом на основі щільності, є методом на основі щільності, особливо ефективним, коли кластери мають довільну форму, а обробка шуму чи викидів є надзвичайно важливою.

Після кластеризації даних продуктивність алгоритму вимірюється за допомогою різних методів перевірки: внутрішньої, заснованої на таких показниках, як бали силуету, або зовнішньої, коли результати порівнюються з відомими мітками або контрольними показниками. Слід, однак, усвідомлювати, що кластеризація є технікою без нагляду, і результати іноді можуть бути нечіткими; таким чином, інтерпретація кластерів іноді може бути суб’єктивною. Тому знання предметної області значною мірою сприяє інтерпретації значущості кластерів.

Кластеризація є потужною для виявлення шаблонів і структури в даних; додатків багато. Оскільки вона може групувати дані без будь-яких зовнішніх міток, кластеризація є незамінною технікою в дослідницькому аналізі даних, розкриваючи ідеї, які неможливо знайти за допомогою інших засобів аналізу даних. Однак кластеризація виконується з великою варіативністю залежно від правильного вибору алгоритму, якості даних та інтерпретації отриманих кластерів, що підкреслює роль обчислювальних методів і досвіду предметної області в досягненні значущих результатів.

1.2 Ключові підходи до кластеризації в обробці текстових даних

Кластеризація в обробці текстових даних – це процес групування подібних текстових документів або фрагментів тексту на основі їх вмісту для виявлення базових шаблонів або тем у великих колекціях тексту. Враховуючи складність, притаманну природній мові, і природну багатовимірність текстових даних, методи кластеризації розвинулися в цій області, щоб вирішити проблеми, пов’язані як зі структурною складністю мови, так і з проблемами представлення тексту для обчислювального аналізу в значущий спосіб [3]. Основні підходи до кластеризації текстових даних загалом поділяються на дві категорії: ті, що покладаються на традиційні векторні методи, і ті, що використовують більш сучасні, просунуті методи обробки природної мови.

Одним із основних підходів до кластеризації тексту є представлення тексту в числовій формі за допомогою таких методів, як модель сумки слів або TF-IDF. Модель сумки слів спирається на векторне представлення документів у просторі великої розмірності, де кожен певний вимір відповідає певному слову всього корпусу. Зазвичай значення кожного параметра вказує на частоту використання слова в цьому документі. Хоча це досить просто й ефективно, основним недоліком є те, що в ньому відсутні будь-які ознаки порядку слів і втрачається значна частина синтаксичної та семантичної інформації. Щоб вирішити цю проблему, TF-IDF покращує це шляхом зважування частоти слів у документі за їхньою зворотною частотою в усьому корпусі. Це налаштування слугує для пом’якшення потужності загальних слів, таких як «the» або «and», які часто з’являються в усіх документах, і робить представлення більш чутливим до слів, які відрізняють окремі документи.

Хоча ці традиційні методи все ще є досить фундаментальними, вони мають обмеження для розуміння глибших семантичних зв’язків між словами та документами [4]. Таким чином, нещодавно з’явилося поширення більш

просунутих методів, які включають техніку вбудовування слів. Серед них Word2Vec, GloVe та fastText створюють представлення слів у щільних векторах у безперервному векторному просторі, де семантично подібні слова розташовуються ближче одне до одного. Зазвичай ці вбудовування вивчаються з великих корпусів у абсолютно неконтрольований спосіб, моделюючи контекстуальні зв’язки слів за допомогою шаблонів спільного входження. Після того, як слова спроектовано на безперервні простори, кластеризація може бути виконана шляхом агрегування вбудованих слів у представлення на рівні документа, а потім застосування до них алгоритмів кластеризації. Ці методи мають перевагу вловлювати нюанси значень слів та їх контекстуальні варіації, отже, забезпечуючи більш багаті представлення для кластеризації порівняно з дискретними, високовимірними векторами в підході сумки слів.

Але окрім вбудовування слів, є новіші розробки в глибокому навчанні, які допомагають покращити кластеризацію текстових даних. Попередньо підготовлені мовні моделі, включно з BERT, що спочатку розшифровується як Bidirectional Encoder Representations від Transformers, мали революційний вплив на обробку тексту, забезпечуючи контекстно-залежні вбудовування, які розглядають решту слів для представлення окремих слів або всього документа. Архітектури таких моделей на основі трансформаторів створюють контекстні вбудовування, тобто представлення кожного слова змінюється залежно від контексту в реченні. Ця здатність створювати контекстуалізовані представлення дозволяє методам кластеризації отримати значну потужність, оскільки тепер семантичні зв’язки між словами можна вловлювати набагато краще. Тонке налаштування цих моделей на тематичних текстах або включення їх у кластеризацію дає більш точні та значущі групи текстових даних.

Різні алгоритми кластеризації для обробки текстових даних включають k-середні, ієрархічну кластеризацію та DBSCAN. Часто їх вибір керується

характером текстових даних і метою кластеризації. K-means – це широко використовуваний алгоритм для неконтрольованої класифікації, метою якого є досягнення заданої кількості кластерів із забезпеченням мінімальної дисперсії всередині кластера. Він ефективний тільки в тому випадку, якщо очікується, що кластери будуть сферичними і приблизно однаковими за розміром [4, 5]. Однак k-середні чутливі до початкового розташування центроїдів кластерів і борються з кластерами неправильної форми або незбалансованими. Ієрархічна кластеризація є більш гнучкою, не вимагає апріорної кількості кластерів, і вона створює дерево кластерів шляхом їх рекурсивного злиття або розбиття на основі певної метрики відстані. Це особливо корисно, коли зв'язки між кластерами є ієрархічними або якщо кількість кластерів, які мають бути сформовані, невідома. З іншого боку, DBSCAN – це алгоритм на основі щільності, який знаходить кластери на основі щільності точок даних, що дає йому змогу бути стійким до шуму та виявляти кластери довільної форми. Це стає особливо ефективним у візуалізаціях, де або дані містять викиди, або, знову ж таки, щільність у наборі даних змінюється.

Як правило, результати кластеризації важче оцінити під час обробки тексту порівняно з числовими даними, оскільки основні мітки істинності можуть не надаватися. Для вимірювання продуктивності кластеризації можна застосовувати такі методи, як силуетні бали, які обчислюють згуртованість кластера та відокремлення від інших кластерів, або внутрішні показники, такі як сума квадратів усередині кластера. Однак у випадку текстових даних предметно-спеціальна інтерпретація кластерів і перевірка стають критично важливими для семантичної узгодженості та інтерпретованості отриманих груп. Нарешті, кластеризація текстових даних почала еволюціонувати від кількох базових векторних підходів, таких як bag-of-words, TF-IDF, до найсучасніших підходів шляхом вивчення репрезентації та глибоких моделей мови. Ці новіші підходи охоплюють семантичне багатство мови, що дозволяє

краще кластеризувати текст [5, 6]. У поєднанні з відповідними алгоритмами кластеризації ці методи дозволяють витягувати значущі структури з великих і складних текстових наборів даних, починаючи від категоризації документів до моделювання тем і аналізу настроїв. Постійне вдосконалення підходів до представлення тексту та кластеризації робить поле безмежним, оскільки постійно винаходяться нові та більш складні способи організації та аналізу текстової інформації.

1.3 Різні види кластеризації: жорстка та м’яка кластеризація

Кластеризація – це техніка неконтрольованого навчання, яка знаходить шаблони або структури в даних шляхом поділу даних на групи, що складаються з подібних об’єктів. Серед найважливіших концепцій кластеризації є різниця між жорсткою та м’якою кластеризацією, двома парадигмами, що визначають, як точки даних призначаються кластерам і як обробляються межі між кластерами. Хоча обидва націлені на розділення даних, обидва методи значно відрізняються через те, як вони обробляють неоднозначність, а також зв’язки однієї точки з декількома або одними кластерами.

Жорстка кластеризація – це підхід, при якому кожен елемент даних повинен бути однозначно призначений певному класу. Тобто в результаті для кожного елемента в наборі даних алгоритм видає чітке, жорстке та неперекриваюче членство в кластері. Стандартним прикладом жорсткої кластеризації є k-середнє, яке для заданої кількості кластерів призначає всі точки даних кластеру найближчого центроїда. У жорсткій кластеризації метою є мінімізація дисперсії в кожному кластері шляхом повторного переміщення центроїдів до досягнення конвергенції. Це вид підходу, який слід використовувати, коли кластери компактні й не об’єднані з іншими. Завдяки своїй простоті він знаходить застосування в сегментації зображень для

категоризації документів і вимагає мінімальних обчислювальних витрат. Як уже зазначалося, існують очевидні обмеження жорсткої кластеризації щодо неоднозначності, властивої деяким наборам даних, і розмитих меж між розділами даних [7]. У таких випадках жорстка природа жорсткої кластеризації може призвести до поганого розподілу кластерів або нездатності вловити тонкощі даних. Наприклад, якщо два кластери мають деякі спільні характеристики, жорстка кластеризація буде довільно призначати точки даних, які знаходяться поблизу межі, одному або іншому кластеру неправильно представляючи справжню структуру даних. Це може призвести до поганої адаптації жорсткої кластеризації до складних, зашумлених або багатовимірних даних.

М’яка кластеризація є дуже гнучкою, оскільки передбачає можливість позиціонування кожної точки даних у групах одночасно з певною приналежністю або ймовірністю, яка відноситься до певного кластера: Замість того, щоб приймати жорсткі рішення щодо членства в кластері, м’яка кластеризація забезпечує міру невизначеності, яка відображає той факт, що точка даних може мати спільні характеристики з більш ніж одним кластером. Алгоритми м’якої кластеризації включають добре відомий алгоритм FCM, який призначає кожній точці даних значення приналежності для кожного кластера, щоб сума всіх цих значень приналежності для кожної точки даних дорівнювала одиниці. Ці значення належності представляють ймовірність того, що дана точка даних належить до певного кластера; вони оновлюються ітеративно на основі відстаней між точками даних і центроїдами кластерів [8].

М’яка кластеризація дуже корисна у випадках, коли між кластерами немає чітких меж або коли характеристики набору даних мають тенденцію збігатися. Наприклад, під час обробки природної мови слова чи документи можуть мати змішані теми чи теми, і м’яка кластеризація може краще вловити ці зв’язки. Крім того, сегментація зображення може виділити поступові переходи між різними класами, факт, який часто спостерігається в реальних

зображеннях, наприклад градація кольору або текстури. Дозволяючи часткове членство в кластері, м’яка кластеризація пропонує багатший і гнучкіший перегляд даних, що може призвести до більш точних і інтерпретованих результатів зі складними наборами даних.

Незважаючи на ці переваги, м’яка кластеризація також приносить ряд проблем. Найважливіша проблема полягає в тому, що інтерпретація значень членства іноді може бути менш інтуїтивно зрозумілою порівняно з прямими призначеннями в жорсткій кластеризації. Крім того, м’яка кластеризація може бути більш вимогливою до обчислень, оскільки підтримка ряду значень членства для кожної точки даних і їх оновлення може значно ускладнити процес кластеризації, особливо для великих наборів даних.

Різниця, по суті, полягає в природі призначення кластерів: жорстка кластеризація призначає кожну точку даних одному кластеру з певним членством, і це підходить для чітко визначених даних, що не перекриваються. М’яка кластеризація дозволяє більш гнучко призначати, де кожна точка даних може певною мірою належати більш ніж одному кластеру; таким чином, він може вмістити більш складну та невизначену структуру даних [8, 9]. Жорсткий і м’який підходи мають ряд відносних переваг і недоліків; Жорстка кластеризація ефективна для гострих випадків, тоді як м’яка кластеризація забезпечує гнучкість і тонкість у наборах даних із неоднозначними або м’якими межами між кластерами. Який підхід вибрати між жорстким і м’яким, залежить від характеру наявних даних і мети, з якою потрібно виконати кластеризацію.

1.4 Метрики схожості для текстових даних

Показники подібності текстових даних є важливими в ряді завдань НЛП, оскільки вони дозволяють кількісно визначити подібність між двома текстовими документами, фразою чи навіть словом. Ці показники дозволяють

багатьом програмам, таким як кластеризація документів, пошук інформації та класифікація тексту, оцінювати, наскільки два тексти схожі, надаючи їм базу. Більшість поточних зусиль, спрямованих на вирішення цих завдань, значною мірою залежать від ефективності та точності вибору метрики подібності, оскільки різні метрики висвітлюють різні аспекти тексту, починаючи від лексичної подібності, семантичного значення до синтаксичної структури. Загалом існують метрики подібності, які стосуються точного збігу між текстовими елементами, включаючи слова чи фрази, та іншими більш абстрактними рівнями, що включають представлення, наприклад, значення слова або значення речення [10].

Найбільш поширеною метрикою, яка є дуже корисною в ряді додатків, що порівнюють текст за допомогою представлень векторів у просторі великої розмірності, є косинусна подібність. Він вимірює косинус кута між двома векторами слів, що представляють відповідні документи або терміни. У тексті вони зазвичай генеруються з уявлень TF-IDF, за допомогою яких документ перетворюється на вектор частотою слів у документі, скоригованою з урахуванням їх важливості в усьому корпусі. Косинусна подібність коливається від 0 до 1, де значення 1 вказує на те, що два вектори є ідентичними, або тексти абсолютно подібні, а 0 означає ортогональність, тобто тексти не мають спільних термінів [11]. Перевага косинусної подібності полягає в тому, що вона враховує напрямок векторів, а не величину, що робить її особливо корисною для порівнянь документів, у яких абсолютна частота термінів менш важлива, ніж їх відносний розподіл.

Іншим дуже поширеним показником подібності є подібність Жаккара, яка базується на ідеї порівняння перетину двох множин. У застосуванні до текстових даних ці набори складаються зі слів або термінів, наявних у документах. Індекс Жаккара визначається як розмір перетину множин, поділений на розмір їхнього об’єднання, що повертає оцінку подібності між 0 і 1. Подібність Жаккара працює особливо добре, коли наявність або

відсутність певних слів важливіша для визначити схожість, ніж частотність слів. Цей захід часто застосовують у завданнях класифікації тексту та кластеризації, де метою є групування документів на основі схожого вмісту за спільними ключовими словами. З іншого боку, деякі з обмежень подібності Жаккара є, коли є справу з довшими або складнішими текстами, оскільки не враховуємо зв'язки між словами-синонімами чи семантичне значення [10, 11].

Для тонкого текстового аналізу використовуються показники семантичної подібності, які намагаються охопити глибше значення слів і фраз, а не просто поверхневе накладання. Загальні представлення включають вбудовування слів, коли слова відображаються в безперервні векторні простори за допомогою таких моделей, як Word2Vec, GloVe та fastText. Ці вкладення розміщують семантично схожі слова ближче одне до одного у векторному просторі; отже, метрики подібності, засновані на таких вставках, враховуватимуть значення слів у контексті. Вбудовування слів часто використовується разом із косинусною подібністю для вимірювання косинусного кута між векторами слів або документів щодо їх подібності. Особливо цінний для програм, таких як машинний переклад, аналіз настроїв і системи відповідей на запитання, цей підхід дає більше розуміння, ніж просто наявність або відсутність слів. Вбудовування слів разом із відповідними показниками подібності може впоратися з полісемією – наявністю кількох значень для одного слова – та синонімією; отже, вони набагато сильніші у вловлюванні тонкощів мови порівняно зі старомодними уявленнями слів [12].

Іншим важливим показником подібності є евклідова відстань, міра прямолінійної відстані між двома векторами в багатовимірному просторі. Хоча евклідова відстань широко використовується в багатьох областях машинного навчання, вона менш ефективна, ніж косинусна подібність для текстових даних, оскільки на неї впливає величина векторів. Це може призвести до недоліку в тому, що в текстових завданнях документи різної довжини призводять до великих відмінностей щодо евклідової відстані, навіть

якщо їхній вміст подібний [11, 12]. Часто методи нормалізації допомагають вирішити цю проблему; все-таки евклідова відстань є менш сприятливою, ніж інші щодо подібності тексту, особливо коли фактор частоти відіграє вирішальну роль у порівнянні.

Іншим показником, який часто використовується для визначення різниці між двома рядками, є відстань редагування, яка враховує мінімальну кількість операцій, які необхідно виконати для перетворення одного рядка в інший. Ці операції зазвичай включають вставки, видалення та заміни символів. Хоча відстань редагування була дуже корисною для порівняння менших рядків, таких як слова чи речення, пряме застосування для більших текстів може бути дорогим з точки зору обчислень і відсутності фіксації ширшої семантичної подібності в тексті. У будь-якому випадку з’являються деякі корисні програми для перевірки орфографії, нечіткої відповідності та виправлення помилок.

Більш конкретна метрика, яка фіксує семантичну подібність у текстових даних, – це використання вбудованих речень або документів, які зазвичай походять від BERT (двонаправлені представлення кодувальника від Transformers). На відміну від традиційних вбудованих слів, які представляють окремі слова, моделі на основі BERT створюють контекстні вбудовані для повних речень або документів. Ці вкладення виводяться з урахуванням не лише значення слів у реченні, але й їхніх синтаксичних зв’язків і контексту, у якому вони трапляються [13]. Косинусну подібність або інші міри відстані можна застосувати до цих вбудованих документів, щоб оцінити загальну подібність між документами як з точки зору використовуваних слів, так і їх значення в контексті. Це особливо корисно для завдань, які вимагають глибокого семантичного розуміння тексту, включаючи семантичний пошук, резюмування та пошук документів, у випадках, коли зміст, що лежить в основі змісту, є суттєвим.

Зрештою, набір показників подібності для текстових даних надає різноманітний набір інструментів для порівняння та аналізу текстового вмісту.

Від простих вимірювань косинуса та подібності Жаккара до більш складних підходів, що включають вбудовування слів і моделі глибокого навчання, ці показники є основою багатьох завдань обробки природної мови. Кожна з цих метрик має сильні та слабкі сторони, тоді як зазвичай вибір серед них обумовлений вимогами завдання, включаючи, наприклад, рівні абстракції, характер аналізованих даних або обчислювальні можливості. У той час як обробка природної мови продовжує розвиватися, можна також очікувати, що нові та більш складні метрики для вимірювання подібності так само продовжуватимуть покращувати наші поточні можливості справлятися зі складним багатством людської мови.

1.5 Огляд сучасної літератури

Стаття [1] описує метод текстового кластеризації, який використовує безкероване навчання для групування текстових документів за подібністю. У статті пропонується новий підхід до кластеризації текстів на основі оптимізації бактеріальної колонії (BCO), який розв’язує проблеми традиційних методів, таких як локальні оптими, низька швидкість збіжності та точність. Автори проводять експерименти на трьох різних наборах даних текстових документів, оцінюючи ефективність підходу за допомогою трьох різних показників. Результати показують, що новий підхід на основі BCO забезпечує високу точність і швидку збіжність. Також проводиться порівняння з іншими методами кластеризації для оцінки сили і надійності запропонованої стратегії.

Стаття [2] присвячена кластеризації текстових даних про промислові несправності, які є специфічним видом коротких текстів, що містять записи про несправності на заводах. Кластеризація таких даних дозволяє зменшити надлишковість і виявити приховану інформацію, що є важливим для підвищення ефективності використання цих даних. Оскільки ці дані є

неструктурованими і нерегулярними, кластеризація стикається з численними викликами. У статті розглядаються існуючі алгоритми кластеризації коротких текстів і вказуються їх недоліки. Зазначено, що основною проблемою є суперечність між вимогами та налаштуванням параметрів, що призводить до низької точності при кластеризації корпусів різних розмірів. Для підвищення точності кластеризації запропоновано вдосконалений алгоритм, який здатен розв'язати цю суперечність. Результати порівняльних експериментів показують, що цей покращений алгоритм має кращу продуктивність порівняно з DBSCAN на корпусах різних розмірів при кластеризації текстових даних про промислові несправності.

Стаття [3] присвячена проблемі кластеризації коротких текстів, яка полягає в безкерованому групуванні подібних коротких текстових документів або одиниць. Оскільки короткі тексти використовуються в багатьох додатках, ця проблема стає все більш актуальною. Основними цілями будь-яких алгоритмів кластеризації даних є висока однорідність і повнота кластерів, однак для коротких текстів ці цілі важко досягнути через те, що ці дані часто представляються розрідженими векторами в просторах з дуже високою розмірністю. У статті запропоновано алгоритм кластеризації VEPHC, що складається з двох етапів, для подолання проблеми розрідженості та високої розмірності коротких текстів. На першому етапі (VEP) початкові векторні представлення проєктуються в простір з меншою розмірністю за допомогою побудови та оцінки змінних комбінацій ознак (термінів). На другому етапі (HC) алгоритм покращує однорідність і повноту кластерів через операції поділу та злиття, що базуються на подібності елементів між кластерами. Експериментальна оцінка VEPHC на двох реальних наборах даних показала його перевагу над іншими сучасними алгоритмами кластеризації за показниками F1-оцінки та нормованої взаємної інформації.

Стаття [4] описує використання текстової майнінг-аналітики для визначення популярних бізнесів в Індонезії, використовуючи дані з кількох

новинних порталів країни. Для цього застосовуються методи попередньої обробки текстів, які перетворюють заголовки новин та теги в числові ваги, що дозволяють класифікувати дані. Ваги потім обробляються за допомогою алгоритму K-Means для групування даних у кластери, і кожен кластер візуалізується за допомогою Word Cloud, що дозволяє ідентифікувати популярні слова. Для оцінки якості кластеризації використовується коефіцієнт Силуета, що дозволяє оцінити наскільки добре кожен елемент відповідає своєму кластеру. Аналіз проводиться щомісяця протягом 2018 року, використовуючи загалом 995 даних, з середнім числом 6 кластерів на місяць. У січні найбільш популярним бізнесом стала торгівля шкіряними сумками, що виявилось за результатами аналізу кластеру з 64 даними, де кластер 1 мав найбільшу кількість учасників, а результати тесту Силуета показали високу якість кластеризації для більшості елементів.

Стаття [5] розглядає зростаючу залежність людей від комп'ютерів та потужностей, які вони надають, для обробки великих обсягів даних, що генеруються щодня. Для аналізу таких даних використовуються кластерні комп'ютери, і кластеризація даних виявилася корисним підходом у майнінгу даних. У цій роботі застосовано алгоритм Apriori, один з найбільш використовуваних алгоритмів для видобутку частих наборів елементів з великих даних, за допомогою кластеру Raspberry Pi. Метою є створення кластеру для забезпечення можливостей аналізу даних, що демонструє потенціал кластерних обчислень та їх застосування у аналітиці даних. Кожен Raspberry Pi використовує стандарт MPI та multiprocessing Python для поділу великого завдання та координації результатів серед групи з чотирьох або більше систем MPICH. Особливу увагу приділено проблемі балансування навантаження на етапі поділу даних. Результати тестування показують, що кластеризація пришвидшує послідовну класифікацію в 10 разів. Також було помічено суттєве покращення продуктивності при додаванні додаткових

процесорів, а також те, що кількість елементів має більший вплив на ефективність кластеризації, ніж кількість транзакцій.

Стаття [6] розглядає зростаючу залежність людей від комп'ютерів та потужностей, які вони надають, для обробки великих обсягів даних, що генеруються щодня. Для аналізу таких даних використовуються кластерні комп'ютери, і кластеризація даних виявилася корисним підходом у майнінгу даних. У цій роботі застосовано алгоритм Apriori, один з найбільш використовуваних алгоритмів для видобутку частих наборів елементів з великих даних, за допомогою кластеру Raspberry Pi. Метою є створення кластеру для забезпечення можливостей аналізу даних, що демонструє потенціал кластерних обчислень та їх застосування у аналітиці даних. Кожен Raspberry Pi використовує стандарт MPI та multiprocessing Python для поділу великого завдання та координації результатів серед групи з чотирьох або більше систем MPICH. Особливу увагу приділено проблемі балансування навантаження на етапі поділу даних. Результати тестування показують, що кластеризація пришвидшує послідовну класифікацію в 10 разів. Також було помічено суттєве покращення продуктивності при додаванні додаткових процесорів, а також те, що кількість елементів має більший вплив на ефективність кластеризації, ніж кількість транзакцій. Ця стаття присвячена методу автоматичної категоризації користувачів на основі даних про їх транзакції на електронних торгових платформах, що дозволяє аналізувати їх покупательські поведінки та споживчі звички. Важливо для таких галузей, як реклама та маркетинг, виявляти певні групи користувачів для таргетингу. У роботі запропоновано новий метод категоризації, який базується на інтересах користувачів до продуктів. Цей метод вводить нову метрику відстані для порівняння користувачів і використовує ефективний алгоритм кластеризації на її основі. Експерименти проведено на даних з вебсайту Yelp, де протестовано набір даних з 1,2 мільйона користувачів. Порівняно з двома іншими популярними алгоритмами кластеризації, запропонований метод

демонструє найкращу ефективність і є підходящим для великих наборів даних. Крім того, користувачів у кожному кластері узагальнюють за допомогою ключових слів, що витягуються з їх відгуків.

Стаття [7] розглядає проблему обробки великих потоків даних з високою розмірністю, де зберігання всього набору даних є непрактичним, тому необхідно використовувати спеціалізовані структури даних для інкрементального узагальнення вхідного потоку. Проблема ускладнюється, коли мова йде про високорозмірні текстові дані через їх високу розрідженість. У статті запропоновано новий підхід до топологічного безкерованого навчання для потоків високорозмірних текстових даних. Цей метод одночасно навчає представлення потоку та кластеризує дані в просторі з меншою розмірністю. Оцінка запропонованого підходу OTTC (Online Topological Text Clustering) і порівняння з сучасними методами здійснюється за допомогою програмного середовища MOA (Massive Online Analysis), яке є відкритим програмним забезпеченням для оцінки потокових даних. Результати показують, що запропонований метод перевищує класичні методи і демонструє багатообіцяючі результати для кластеризації потокових високорозмірних текстових даних.

Стаття [8] присвячена проблемам кластеризації текстів і використанню технології оптимізації рою частинок (PSO), зокрема необхідності визначення відповідної кількості кластерів перед кластеризацією, чутливості результатів до початкового центру кластерів та дефектів PSO, таких як передчасна збіжність і схильність до локальних оптимумів. Для вирішення цих проблем запропоновано алгоритм DTA-PSO (Dynamic Topology Adaptive PSO), який поєднує алгоритм K-means та PSO для автоматичної кластеризації текстів. У DTA-PSO застосовується вдосконалена стратегія динамічного навчання, що базується на досвіді соціальної групи, де джерелом соціального досвіду для частинок є не тільки найкраща частинка серед усіх сусідів, а й враховується їх ефективність і відносне положення з поточною частинкою для розрахунку

цільової позиції на наступній ітерації. Для експериментів використовуються три набори даних з відкритих джерел, зокрема частина китайського корпусу Університету Фудан. Результати численних експериментів показують, що алгоритм DTA-PSO покращує визначення значення K та ефективність кластеризації порівняно з традиційними методами.

Стаття [9] пропонує новий підхід до класифікації новин, який інтегрує ієрархічну кластеризацію та ансамблеві алгоритми. Спочатку зібрано та попередньо оброблено набір даних BBC News з Kaggle, де текстові дані були піддані токенізації, видаленню стоп-слів та векторизації за допомогою TF-IDF. Потім застосовувалась ієрархічна кластеризація для групування схожих документів за змістом за допомогою алгоритму агломеративної кластеризації, при цьому кількість кластерів була регульованою для досягнення різної деталізації. Отримані мітки кластерів були присвоєні відповідним документам, і з кластеризованих даних були витягнуті додаткові ознаки. Ці ознаки разом з матрицею TF-IDF стали входом для навчання класифікаторів, таких як Gradient Boosting, Bagging Classifier та Random Forest. Класифікатори були навчені для передбачення категорій документів на основі текстових ознак і міток кластерів. Потім навчальні моделі були протестовані на окремому тестовому наборі для оцінки їх здатності до узагальнення. Результати кластеризації та класифікації ретельно оцінювались для визначення ефективності методу та вивчення взаємозв’язку між кластеризацією документів і продуктивністю класифікації.

Стаття [10] присвячена кластеризації як необхідному методу попередньої обробки даних у дослідженнях з майнінгу даних, який дозволяє виявити внутрішню структуру розподілу цінних наборів даних з нелабельованих даних, що спрощує їх опис. Технологія майнінгу даних дозволяє відкривати потенційну та цінну інформацію з великої кількості даних, надаючи новий зміст масивам даних, накопиченим людьми в епоху інформації. Зі швидким розвитком технології майнінгу даних, кластеризація

за допомогою сіток набуває популярності у аналізі даних, зокрема зображень. Як одна з основних методів майнінгу даних, кластерний аналіз полегшує обробку даних у машинному навчанні, застосовуючи алгоритми для аналізу даних та правильної роботи машин. Завдяки постійним експериментам і доказам попередніх поколінь, алгоритми кластеризації стають дедалі більш досконалими. У статті наведено огляд еволюції високоякісних алгоритмів кластеризації, щоб краще зрозуміти їх застосування та дослідити різні підходи до кластеризації. Вивчено 16 відомих наукових статей, що висвітлюють розвиток кластеризації в різних сферах. Метою цього систематичного огляду є підсумок методів кластерного аналізу та групування, що використовуються до сьогодні, а також рекомендації щодо майбутнього розвитку цієї технології.

Стаття [11] присвячена вирішенню проблеми аналізу високорозмірних даних, зокрема в контексті даних, що мають ієрархічну структуру, як це спостерігається у таких реальних даних, як генетичні дані людини. Автори пропонують алгоритм для визначення кількості груп ознак у високорозмірних даних за допомогою методу послідовного мінімакса та виявлення ієрархічної структури цих даних. Алгоритм дозволяє виявити кілька можливих варіантів групування ознак та їх ваги для кожного числа груп. Після порівняння різних варіантів групування ознак виявляється багатошарова структура груп ознак. Для оцінки ефективності кількості груп ознак та забезпечення методу кластеризації підпросторів запропоновано алгоритм навчання латентних груп ознак (LFGL). Експерименти на декількох наборах генетичних даних показують, що запропонований алгоритм перевершує кілька інших представницьких алгоритмів.

Стаття [12] присвячена дослідженню класифікації намірів у чат-ботах, орієнтованих на виконання певних завдань. Класифікація намірів є складовою текстової класифікації, що вимагає наявності міток у даних для успішного виконання процесу класифікації. Для пришвидшення аналізу висловлювань використовується метод кластеризації, зокрема, кластеризація на основі

щільності, де одним з алгоритмів є DBScan. У дослідженні використано 10 000 висловлювань клієнтів з електронної комерції на платформі WhatsApp. Як інструмент для вбудовування речень застосовано сучасну модель SentenceBert. Найкращий результат кластеризації, з оцінкою силуету 0.327, був отриманий при значеннях eps 0.1 і MinPts 95. Проте за результатами кластеризації виявилося, що деякі речення, позначені як шуми, можуть бути додатково кластеризовані. Для підвищення ефективності кластеризації можна дослідити методи попередньої обробки тексту, аугментації тексту та техніки вбудовування речень.

Стаття [13] пропонує новий підхід до кластеризації коротких текстів, який вирішує обмеження існуючих алгоритмів, що ґрунтуються на одній точці зору тексту, таких як моделі на основі тем чи глибинні моделі кластеризації. Існуючі алгоритми використовують або представлення тексту у вигляді мішка слів (bag-of-words), або вбудовування документів (document embeddings), що обмежує їх ефективність. Для подолання цих обмежень автори пропонують модель Multi-View Clustering (MVC), яка об’єднує обидва підходи. Вони використовують модель Dirichlet Multinomial Mixture (DMM) для представлення текстів як мішка слів і модель Gaussian Mixture Model (GMM) для представлення текстів як вбудовувань. За допомогою випадкової змінної Бернуллі контролюється взаємодія цих двох моделей, що дозволяє моделі MVC використовувати семантичну інформацію вбудовувань текстів та інформацію з мішка слів одночасно. Результати експериментів на чотирьох різних наборах даних показують ефективність цієї моделі.

Стаття [14] присвячена автоматичній кластеризації текстових документів для виявлення окремих груп серед великої кількості неструктурованих документів. Групи документів, що належать до однієї категорії, мають високий ступінь подібності між собою та суттєву відмінність від документів в інших групах. Критерій групування визначається через функцію, значення якої повинно бути максимізоване або мінімізоване. У статті

запропоновано новий метод для обчислення початкових кластерів за допомогою методу фракціонування, а також новий метод вимірювання схожості, який комбінує косинусну міру, з'єднання, сусідів і гауссову функцію, що дозволяє врахувати як локальну, так і глобальну схожість. Описано апроксимований алгоритм кластеризації, який намагається подолати недолік алгоритму k-середніх, зокрема невизначеність кількості ітерацій, фіксуючи їх кількість без втрати точності. Запропонований підхід використовує сусідів і з'єднання разом з гаусовою функцією для пошуку початкових кластерів у задачі кластеризації текстових документів. Експериментальні результати на спортивних документах показують, що ефективність запропонованої техніки є дуже обнадійливою.

Стаття [15] розглядає новий метод текстового майнінгу для системи операційних квитків диспетчеризації, що містить актуальну інформацію про енергетичну систему. Метою є покращення управління диспетчеризацією для енергетичних систем за допомогою алгоритму спектральної кластеризації на основі розділення графів. Квитки замовлень класифікуються за їх змістом на різні типи, і схожі квитки зберігаються в одній групі. Між цими групами моделюється і аналізується взаємозв'язок, і для досягнення кращих результатів кластеризації використовується біпартитний граф. Алгоритм спектральної кластеризації використовує матрицю Лапласа для сегментації графа, отримуючи векторні характеристики матриць і дані для обробки. Після цього отримується результат кластеризації, що відображає процес текстового майнінгу операційних квитків для диспетчеризації енергетичної системи в реальному часі. Результати симуляцій показують, що запропонований метод ефективно та точно аналізує операційний зміст процесу ремонту та управління енергетичними системами, відповідаючи вимогам регулювання роботи енергетичних систем.

Стаття [16] розглядає проблему кластеризації коротких текстів, що є важливим методом аналізу текстів для вилучення знань із онлайн соціальних

медіа, таких як Twitter, Facebook та Weibo. Короткі тексти мають певні складнощі для кластеризації через такі особливості, як абревіатури та неформальні вирази, а також обмежену довжину текстів. Однак тексти, що належать до однієї теми, часто містять спільні терміни або їхні стеми, які можуть ефективно представляти цю тему. Стаття пропонує новий метод виявлення таких представницьких термінів для теми, який допомагає кластеризувати короткі тексти, об’єднуючи їх в групи за найбільш схожими термінами. Метод TRTD вимірює близькість термінів через їх спільну співзалежну появу, а значимість термінів визначається їх глобальними частотами в корпусі текстів. Експериментальні результати на реальних наборах даних показують, що метод TRTD забезпечує кращу точність і ефективність у порівнянні з сучасними методами кластеризації коротких текстів.

1.6 Висновки до першого розділу

Метою цього розділу є узагальнення результатів попередніх обговорень і подальше обговорення наслідків методів кластеризації в цілому, їх практичне застосування, обмеження та поточні розробки. Кластеризація загалом – це тип процедури неконтрольованого навчання, метою якої є поділ даних на групи з подібними характеристиками.

Кластеризація залишається необхідним інструментом аналізу даних, надаючи багату інформацію про структуру та зв’язки в складних даних. На практиці успіх алгоритму кластеризації залежить від розумного вибору між жорсткою та м’якою кластеризацією, правильного вибору показників подібності та ретельного розгляду базових даних. Оскільки домен продовжує розвиватися, запровадження ще більш складних методів, таких як контекстуальне вбудовування та кластеризація на основі щільності, дозволить кластеризації вийти за рамки та стати ще потужнішим інструментом для

аналізу текстових даних. Однак такі проблеми, як усунення шуму, оцінка якості кластерів і масштабування до великих наборів даних потребуватимуть продовження досліджень та інновацій. Майбутнє обробки текстових даних у кластеризації насправді полягає в обробці динаміки та багатоаспектних аспектів людських мов, що відкриє нові шляхи для людського знання та аналізу в цифровому сховищі, що постійно розширюється.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ СУЧАСНИХ АЛГОРИТМІВ КЛАСТЕРИЗАЦІЇ ДЛЯ ГРУПУВАННЯ ТЕКСТОВИХ ДАНИХ

2.1 Алгоритм K-середніх

Основний варіант K-середніх – це, по суті, ітераційний алгоритм, який намагається мінімізувати дисперсію в кожному кластері. Це робиться шляхом призначення кожної точки даних кластеру на основі її близькості до центроїда кластера [2, 4]. Для кластера центроїд зазвичай є «центром». Він обчислюється шляхом обчислення середнього значення всіх точок, які складають певний кластер. У послідовних ітераціях центроїди оновлюються, а точки перепризначаються найближчому центроїду до збіжності алгоритму, де призначення змінюються незначно.

По-перше, коли застосовуються K-середні до текстового набору даних, потрібно визначити кількість кластерів, у цьому випадку K. Це потрібно попередньо оцінити перед фактичним запуском алгоритму, і це часто впливає на якість результату кластеризації. На практиці в K існує експертний досвід, інтуїція та/або система проб і помилок. Наприклад, визначено цільову функцію для кластера новинних статей за кількістю тем, отже, числа можуть видавати фактичне чітке число тем, що існують у певному наборі даних. У багатьох реальних програмах знайти найкращий K може бути принаймні складно, і для оцінки хорошого значення K можна використовувати інші методи, наприклад, метод ліктя або силуетну оцінку.

Як тільки значення K визначено, алгоритм ініціалізує K центроїдів випадковим чином. Ці центроїди можна вибирати різними способами – наприклад, шляхом випадкового вибору K точок даних як початкових центроїдів або використання більш складних методів для покращення процесу ініціалізації та, отже, зменшення ризиків поганої збіжності. Якщо випадкова

ініціалізація потрапляє в локальні оптимуми, для отримання найкращого кластера необхідно виконати кілька алгоритмів із різними ініціалізаціями.

Після визначення початкових центроїдів алгоритм переходить до етапу призначення [5]. На цьому кроці кожна точка даних призначається кластеру, центроїд якого є найближчим до неї. Близькість зазвичай вимірюється метрикою відстані, причому евклідова відстань є найпоширенішою. Також можна використовувати інші міри відстані, наприклад косинусну подібність, яка особливо актуальна для текстових даних. Подібність косинусів забезпечує міру кута між двома векторами і зазвичай є кращою в кластеризації тексту через її нечутливість до величин, а не лише до орієнтації. Це особливо корисно у випадку текстових даних, коли довжина тексту, визначена як кількість слів, може сильно відрізнятися між документами, але часто теми або теми визначаються відносною важливістю деяких слів, а не загальна довжина документа.

Після завершення етапу призначення алгоритм переходить до етапу оновлення, де повторно обчислюються центроїди кластерів. Щоб обчислити новий центроїд кожного кластера, потрібно буде знайти середнє значення всіх точок даних, що входять до нього. Для текстових даних це означало б усереднення між усіма векторами слів у кожному документі, призначеному тому чи іншому кластеру. Ці вектори слів можна отримати з вищих представлень тексту, таких як пакет слів, TF-IDF, або навіть вищих представлень, що включають вбудовування слів, як-от Word2Vec, GloVe або BERT. Ці векторні представлення вловлюють семантичне значення слів, дозволяючи алгоритму кластеризації групувати документи на основі їх вмісту, а не їх конкретних формулювань.

Після обчислення нових центроїдів алгоритм повертається до повторення двох кроків: призначення та оновлення. Ітерації двох кроків тривають до збіжності, тобто центроїди не змінюються суттєво від однієї ітерації до наступної, або досягається певна заздалегідь визначена кількість

ітерацій. Тепер, на цьому етапі, алгоритм розділив текстові дані на K кластерів, де подібні документи в кластері містять однакові теми.

K-means досить простий і ефективний, але він страждає від деяких недоліків, враховуючи текстові дані. Однією з найважливіших проблем є попереднє визначення кількості кластерів, K, заздалегідь. Якщо K занадто малий, можна легко пропустити важливі структурні відмінності в даних; якщо K занадто велике, можуть бути отримані високоспецифічні кластери без суттєвої інтерпретації [13]. Іншим недоліком K-середніх є те, що воно також припускає, що кластери мають сферичну форму та однакові розміри, що не часто трапляється в природних текстових даних реального світу. Наприклад, деякі теми більш різноманітні або мають більш складну структуру, і тому їх неможливо повністю охопити кластеризацією. З огляду на ці та інші питання, було запропоновано кілька варіантів і вдосконалень основного алгоритму K-середніх. Серед іншого, таким варіантом є K-means++, який намагається забезпечити кращу ініціалізацію центроїдів і, таким чином, робить алгоритм більш надійним і менш чутливим до початкового вибору центроїдів. Інший підхід полягає в поєднанні K-середніх з технікою ієрархічної кластеризації, яка дозволяє ідентифікувати найбільш прийнятну кількість кластерів шляхом ініціалізації великої кількості малих кластерів, а потім виконання їх поступового злиття на основі їх подібності. Крім того, пропонуються деякі алгоритми кластеризації на основі щільності, такі як DBSCAN, які не вимагають апріорного визначення кількості кластерів користувачем і можуть обробляти більш загальні форми кластерів.

Незважаючи на всі ці обмеження, K-means залишається одним із найбільш використовуваних і потужних інструментів кластеризації текстових даних. Він простий і ефективний, тому корисний для великих наборів даних. Виявлення прихованих шаблонів у немаркованих даних зробило цю техніку вирішальною в ряді важливих областей машинного навчання: кластеризація документів, тематичне моделювання та аналіз тексту. Це, у свою чергу, буде

корисним для виділення прихованих тем, тенденцій і взаємозв’язків у наборі даних і, отже, надасть дуже цінну інформацію для широкого кола галузей, таких як дослідники, аналітики та практики.

Попередня обробка перед застосуванням K-середніх не менш важлива для текстових даних. Текстові дані часто містять багато шуму, наприклад стоп-слова, знаки пунктуації та спеціальні символи, які можуть заважати процесу кластеризації [14]. Тому перед застосуванням алгоритму зазвичай очищають і попередньо обробляють текстові дані. Це може передбачати видалення стоп-слів (загальних слів, таких як «the» та «and», які не мають істотного значення), нормалізацію тексту (наприклад, перетворення всіх слів на малі літери), а також коріння або лематизацію слів (зменшення їх до кореневої форми).). Інша проблема полягає в тому, що текстові дані зазвичай розріджені, оскільки більшість слів у документі не зустрічаються в інших документах. Хоча ця розрідженість може ускладнити завдання кластеризації, такі методи, як зменшення розмірності, як-от аналіз головних компонентів або декомпозиція сингулярного значення, або використання щільних вставок слів можуть полегшити цю проблему.

Щоб завершити процес кластеризації, потрібно провести оцінку, щоб побачити, наскільки хороша якість кластерів. Оскільки K-means є неконтрольованим алгоритмом, йому не вистачає фактичного або справжнього порівняння його результатів. Отже, неможливо легко оцінити точність кластеризації. Однак існує спосіб визначити якість кластеризації, подібно до оцінки силуету, яка обчислює схожість точки даних із власним кластером порівняно з іншими. Вищий бал силуету означає, що точки добре згруповані, тоді як нижчий вказує на інше, оскільки кластери перекриваються або визначення кластерів погане. Інші показники оцінки включають чистоту, тобто частку документів у кластері, які належать до однієї категорії чи теми, і скоригований індекс Ренду, що порівнює результати кластеризації з відомою основною правдою, якщо така доступна.

Алгоритм K-середніх є основним методом для кластеризації текстових даних. Це простий, але ефективний спосіб розділити документи на групи на основі їх вмісту. Маючи власні недоліки, як і будь-який інший метод, він страждає від того, що вимагає апріорного знання кількості кластерів і чутливості до ініціалізації; це потужна техніка, яка широко використовується в машинному навчанні [14]. За умови відповідної попередньої обробки та правильного вибору вимірювань відстані та K, K-середні можуть дати цінну інформацію з немаркованих текстових даних; отже, K-means є важливим інструментом для кожного дослідника, спеціаліста з обробки даних і спеціаліста, який працює з великомасштабними наборами текстових даних.

2.2 Алгоритм DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) – це один із популярних алгоритмів кластеризації на основі щільності, який широко використовується в галузі машинного навчання для виявлення кластера в даних, особливо складних і неправильних форм. На відміну від класичних алгоритмів кластеризації, наприклад, K-means, який розбиває всі дані на певну кількість фіксованих кластерів, у методі DBSCAN не потрібно мати жодних попередніх знань про точну кількість кластерів. Натомість він використовує засновані на щільності поняття «основних точок», «граничних точок» і «шумових точок» для формування кластерів на основі щільності точок даних у просторі ознак [13, 14]. Алгоритм особливо добре підходить для обробки масивних наборів даних і підтримки різних форм для кластерів, що зробить його особливо корисним у реальних програмах, таких як кластеризація текстових даних.

Основна концепція DBSCAN полягає в тому, що кластер – це область із високою щільністю в точках даних, розділених областями з низькою щільністю. Для досягнення від однієї точки до іншої кластер вважається

групою точок. Двома основними параметрами в цьому алгоритмі, які керують усім процесом кластеризації, є eps і minPts. eps визначає максимальну відстань між будь-якими двома точками, які вважаються сусідами, тоді як minPts визначає кількість точок у групі для формування щільної області. Будь-які дві точки, які існують у радіусі епсилон навколо одна одної, вважаються сусідами, причому центральною точкою є точка, яка має принаймні minPts сусідів у своїй околиці епсилон [15].

Алгоритм DBSCAN починається з ідентифікації основних точок, які є тими точками, які мають достатньо високу щільність оточуючих точок. Точка визначається як основна, якщо на відстані eps є принаймні minPts точок (включаючи саму точку). Після визначення основних точок DBSCAN розширює їх, щоб включити всі точки, доступні з будь-якої з основних точок, утворюючи більші кластери з груп основних точок та їхніх доступних сусідів. Досяжність є транзитивною в тому сенсі, що якщо точка A безпосередньо досяжна з точки B, а точка B безпосередньо досяжна з точки C, то точка A досяжна з точки C.

Точки, недоступні для будь-якої основної точки, а також не можуть утворити кластер із достатньою кількістю сусідів, класифікуються як шумові точки або викиди. Ці точки не призначаються жодному кластеру і, отже, залишаються як шум. Це включало б одну з головних переваг DBSCAN перед іншими методами кластеризації: здатність виявляти та видаляти шумові точки краще справляється з наборами даних, які мають викиди. Також не потрібно призначати всі точки кластерам [16]. Потім алгоритм ітеративно вибирає невідвідані точки з набору даних і формує кластер шляхом розширення навколо основних точок.

З огляду на це, якщо точка стає основною, алгоритм ітераційно розширюється, щоб перевірити, чи існують інші точки, доступні безпосередньо з неї. Якщо це прикордонна точка, то вона повинна бути просто додана до того самого кластера, але сама по собі не є основною точкою. Це

будуть точки, які, перебуваючи на відстані eps, не містять необхідної кількості точок навколо них, щоб стати основними точками. Шум відноситься до таких точок, які ні самі не були основною точкою, ні будь-яка інша точка включала їх під час ідентифікації кластера. Ще однією ключовою особливістю DBSCAN є можливість приймати будь-яку форму або довільну форму кластерів.

Це абсолютно протилежно іншим підходам, таким як K-середні, припускаючи, що кластер має сферичну форму та однакові за розміром. DBSCAN базується на щільності, тому здатний захоплювати кластери, які не є опуклими за формою та дуже різняться за формою та розміром. Це робить DBSCAN дуже корисним для реальних даних, де зазвичай жоден кластер не має форми, що відповідає якійсь простій геометричній інтуїції. Ця функція особливо корисна при роботі з текстовими даними, оскільки теми або теми в тексті часто можуть бути представлені кластерами з неправильними межами. При використанні DBSCAN для текстових даних першим завданням є представлення тексту в якійсь відповідній математичній формі [15].

Текстові дані за своєю суттю є неструктурованими, тому їх не можна використовувати як необроблений текст у алгоритмі кластеризації. Тому його потрібно попередньо обробити до числового формату, який виражає семантичне значення тексту. Це можна зробити за допомогою кількох методів, які включають модель сумки слів, TF-IDF або більш просунуті вбудовування слів, такі як Word2Vec, GloVe та BERT. Ці методи перетворюють текстові документи на вектори у багатовимірному просторі, де кожен вимір відповідає певній функції, наприклад слову або комбінації слів.

Інший важливий аспект застосування DBSCAN до текстових даних стосується вибору міри відстані. Загалом стандартною мірою відстані DBSCAN є евклідова відстань, але для текстових даних більш прийнятною метрикою відстані може бути подібність косинусів, яка розраховує кут між двома векторами, а не їх величину. Косинусну подібність дуже часто використовують у кластеризації тексту, оскільки вона фіксує семантичну

подібність документів, тоді як відмінності в довжині не важливі. Оскільки документи можуть мати великі розбіжності у розмірі, косинусна подібність дозволяє цьому алгоритму приділяти більше уваги вмісту та структурі текстів, а не бути упередженим через їх довжину.

Коли текстові дані представлені у вигляді векторів і вибрано відповідну міру відстані, можна застосувати DBSCAN. Наступним кроком є визначення оптимальних значень для параметрів eps і minPts [16]. Продуктивність DBSCAN дуже чутлива до правильного вибору значень цих параметрів, оскільки вони безпосередньо контролюють кількість і якість сформованих кластерів. Якщо eps занадто великий, алгоритм може об’єднати різні кластери разом. Якщо він занадто малий, алгоритм не зможе розпізнати кластери, і більшість точок буде класифіковано як шум. Таким чином, minPts також потрібно налаштувати відповідно до щільності набору даних. Для текстових даних minPts зазвичай знаходиться в діапазоні від 3 до 10, хоча це може змінюватися залежно від характеру тексту та бажаної деталізації кластерів.

Насправді здатність DBSCAN виявляти кластери довільної форми робить його особливо корисним для кластеризації текстових даних, оскільки структура тем усередині корпусу документів може сильно відрізнятися. Алгоритм DBSCAN здатний знаходити такі нелінійні зв’язки у вигляді щільних областей тексту, сильно взаємопов’язаних у векторному просторі.

Однак це має власний набір недоліків. Однією з ключових проблем є те, що продуктивність цього алгоритму може залежати від вибору параметрів, особливо eps. У деяких сценаріях навряд чи можливо визначити відповідне значення eps, особливо для великих даних, таких як текст. Поширеним методом вибору eps є використання графіка k-відстаней. На графіку k-відстаней для кожної точки в наборі даних відстань до k-го найближчого сусіда відкладається відносно цієї точки. «Лікоть» такого графіка, тобто точка, в якій відстань починає різко збільшуватися, можна використовувати як приблизну оцінку для встановлення eps. Незважаючи на це, вибір eps все ще

може бути нетривіальним і зазвичай вимагає експериментів і знання предметної області.

Іншим обмеженням DBSCAN може бути його обчислювальна складність: оскільки він включає в себе обчислення попарних відстаней, часова складність для дуже великих наборів даних може бути надзвичайною. Особливо це стосується даних великої розмірності, де розмірність може зробити відстань між точками менш інформативною [15]. Щоб пом’якшити це, перед запуском DBSCAN можна застосувати методи зменшення розмірності, такі як аналіз головних компонентів (PCA) або t-розподілене стохастичне вбудовування сусідів (t-SNE), щоб зменшити кількість вимірювань і зробити алгоритм більш придатним для обчислень.

Незважаючи на такі труднощі, DBSCAN залишається ефективним алгоритмом кластеризації текстових даних. Його здатність виявляти кластери довільної форми та обробляти шум робить його універсальним інструментом у багатьох програмах: від кластеризації документів, тематичного моделювання до аналізу тексту. Гнучкість цього алгоритму дозволяє йому використовувати широкий спектр програм, де традиційні методи, такі як k-середні, не дають результатів через припущення щодо сферичних або однакових за розміром кластерів. Виявлення значущої структури в немаркованих наборах даних шляхом групування текстових даних за допомогою DBSCAN на основі локальної щільності може надати важливу інформацію як для дослідників, так і для практиків у різних областях, таких як пошук інформації чи аналіз соціальних мереж.

Порівняння K-means та DBSCAN наведено на рисунку 2.1.

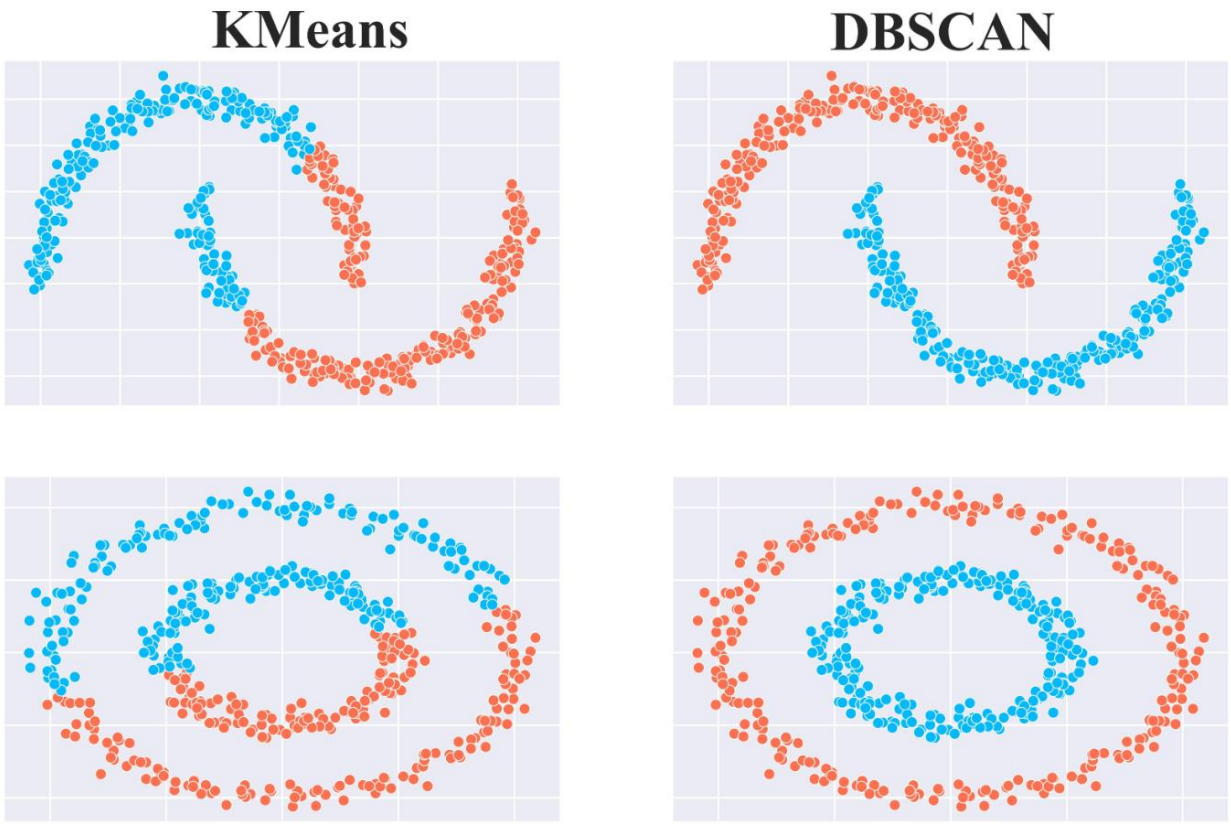

Рисунок 2.1 – Порівняння K-means та DBSCAN

DBSCAN – це потужний алгоритм кластеризації на основі щільності, який виявився дуже успішним на складних масивах даних великої розмірності, таких як текст. Він знаходить кластери довільної форми та справляється з шумом у даних – гарний перепочинок порівняно з багатьма класичними методами кластеризації. Його здатність застосовувати всі види вимірювань відстані дозволяє легко вписати цю технологію в інші контексти порівняно, наприклад, з кластеризацією документів або зображень. Хоча алгоритм має ряд обмежень, таких як чутливість до вибору параметрів і обчислювальна складність, їх часто можна подолати шляхом ретельної попередньої обробки та налаштування параметрів разом із використанням методів зменшення розмірності. DBSCAN є важливим інструментом для кластеризації текстових даних, пропонуючи дуже надійну техніку для виявлення прихованих шаблонів і структур у великих наборах даних без міток.

2.3 Агломеративна кластеризація

Агломеративна кластеризація – це один із популярних алгоритмів ієрархічної кластеризації для групування точок даних на основі їх подібності. Він підпадає під категорію кластеризації знизу вгору, де кожна точка даних починається з власного кластера, а з часом об’єднується з іншими кластерами на основі певних критеріїв подібності чи відмінності [7, 9]. Цей процес об’єднання продовжується ітеративно, доки всі точки даних не стануть частиною одного кластера або поки не буде досягнуто умови зупинки, скажімо, кількох кластерів. Агломеративна кластеризація може бути дуже корисною під час роботи зі складними наборами даних, такими як текст, де зв’язки між точками даних часто відрізняються нюансами та багатогранністю. Цей метод є надійним і надає широку область застосування в аналізі тексту, від кластеризації документів до більш складних завдань, таких як тематичне моделювання.

Основна перевага агломеративної кластеризації полягає в її гнучкості та можливості побудови деревоподібної структури, яка називається дендрограмою. Дендрограма – це деревоподібна діаграма, яка представляє ієрархічні зв’язки між точками даних і відображає структуру набору даних, прихованого за даними. Це покаже для текстових даних, як різні документи пов’язані один з одним і як їх можна згрупувати на основі спільних тем або тем. На відміну від таких алгоритмів, як K-середні, які вимагають попереднього вказівки користувачем кількості кластерів, агломеративна кластеризація дозволяє природним чином з’являтися кількості кластерів під час виконання алгоритму. Це техніка неконтрольованого навчання, яка не вимагає попереднього позначення даних, і, отже, особливо корисна в програмах, де основні групи або категорії невідомі заздалегідь.

У центрі агломеративної кластеризації знаходиться показник подібності або відмінності між точками даних. У випадку з текстом головна проблема

полягає в попередній обробці неструктурованого необробленого тексту в математичне представлення, яке представляє семантичне значення документів. Це включає, наприклад, представлення кожного документа як вектора у багатовимірному просторі [12]. Найпоширенішим способом перетворення цього в числові вектори є модель сумки слів, яка представляє документи як вектори частот слів. Незважаючи на те, що цей підхід є простим, зазвичай йому не вистачає глибшого розуміння тексту та зв'язків між словами. Більш досконалим є термін частотно-інверсна частота документа, який, окрім підрахованих частот слів, враховує важливість кожного слова в документі по відношенню до всього корпусу. Це дуже добре для пошуку важливих слів і фраз у документі та оцінки їх важливості. Більш сучасні методи вбудовують слова в безперервний векторний простір для захоплення семантичного значення, завдяки чому семантично подібні слова розміщуються ближче одне до одного.

Після представлення текстових даних у вигляді вектора наступним кроком буде обчислення міри подібності або відстані між документами. Найпоширеніші вимірювання відстані для агломеративної кластеризації включають евклідову відстань, косинусну подібність або інші показники відстані, придатні для даних великої розмірності. Загалом косинусну подібність є кращою для текстових даних. По суті, він обчислює косинус кута між двома векторами, щоб зафіксувати напрямок, а не їх величину. Це стає дуже ефективним у випадку текстових даних, де довжина документів може бути різною, але основні теми чи теми загалом містяться в тексті, і вони представлені відносною важливістю слів, а не простою їх частотою. Отже, косинусна подібність є показником, який зосереджується на вмісті та не змінюється щодо довжини документа.

Після того, як матриця відстані побудована, агломеративна кластеризація триває шляхом ітераційного об'єднання кластерів. Основна ідея такого об'єднання полягає в об'єднанні на кожному кроці двох найбільш

схожих кластерів. Цей процес повторюється, поки всі точки даних не будуть в одному кластері або поки не буде досягнуто бажаної кількості кластерів. Критерій зв'язку – це показник, який визначає відстань між двома кластерами та визначає, як кластери слід об'єднувати. В агломераційній кластеризації використовується кілька різних типів, кожен з яких по-різному впливає на злиття кластерів [12, 15, 16]:

1) один зв'язок. У цьому методі відстань між двома кластерами визначається як мінімальна відстань між будь-якою парою точок, по одній з кожного кластера. Цей метод, як правило, призводить до витягнутих, схожих на ланцюг кластерів, оскільки алгоритм об'єднує кластери на основі лише однієї точки в одному кластері, яка є проксимальною до точки в іншому кластері;

2) повний зв'язок. Відстань між двома кластерами визначається тут як максимальна відстань між будь-якою парою точок, по одній з кожного кластера. Повне з'єднання створює більш компактні сферичні кластери, оскільки для злиття всі точки в одному кластері повинні бути відносно близькими до всіх точок в іншому кластері;

3) середній зв'язок. Тут відстань між двома кластерами визначається як середня відстань між усіма парами точок з двох кластерів. Середнє з'єднання має тенденцію бути десь між одиночним і повним з'єднанням з точки зору їх тенденцій і часто створює більш збалансовані форми кластерів;

4) зв'язок Уорда. Цей метод об'єднує два кластери шляхом мінімізації збільшення загальної дисперсії всередині кластера. Він спрямований на створення компактних і однорідних кластерів. Зазвичай зв'язування Уорда є кращим, коли потрібно мінімізувати розбіжності в кластерах, що особливо корисно в текстовому кластеризуванні, коли документи в кожному кластері мають бути якомога подібнішими.

Використаний критерій зв'язку може призвести до дуже різних кластерів. Для текстових даних кластери можуть дуже відрізнятися залежно

від того, чи використовується одиночне, повне, середнє зв’язування чи зв’язок Уорда. Фактично, за різними методами зв’язування існують різні типи кластерів, що стосуються їх форм, розмірів і зв’язності. Тому правильний вибір критерію зв’язку вимагає розуміння даних і мети, з якою здійснюється кластеризація.

Однією з помітних переваг агломеративної кластеризації є те, що вона забезпечує ієрархічну структуру, яка забезпечує більшу гнучкість у визначенні кількості кластерів. Дендрограмма, створена в процесі кластеризації, відображає, як утворюються кластери на кожному кроці алгоритму. Користувач може контролювати кількість кластерів у кінцевому результаті, обрізаючи дендрограму на будь-якому бажаному рівні. Це означає, що для агломеративної кластеризації не потрібна кількість кластерів як вхідних даних, на відміну від інших алгоритмів, таких як K-means, які вимагають від користувача вказати K. Дендрограма також дає уявлення про зв’язки між цими кластерами: який із кластери ближче і коли вони приєдналися до поточного в процесі злиття.

Така функціональність для дослідження дендрограми та визначення належного рівня кластеризації у випадку текстових даних має велике значення, коли реальна структура даних невідома. Дендрограмма дозволить аналітикам пограти з номерами кластерів, щоб важливі закономірності в даних можна було охопити остаточним набором кластерів без надмірного або недообладнання. Наприклад, розглянемо завдання кластеризації статей новин за темами. Дендрограмму можна перевірити на предмет точки, де теми справді починають розходитися і, таким чином, природним чином поділяють текстові дані на значущі кластери.

Однією з проблем агломеративної кластеризації, особливо в контексті текстових даних, є обчислювальна складність. Більшість алгоритмів обчислюють попарну матрицю відстаней, яка обчислювально дорога для великих наборів даних, особливо у просторах великої розмірності. Часова

складність агломеративної кластеризації зазвичай становить $O(n^2)$ для n точок даних, оскільки на кожному кроці необхідно порівнювати кожну пару точок даних, щоб оцінити їх подібність. Це може бути особливо проблематично при роботі з великими текстовими корпусами [2, 8]. Однак є способи обійти це. Наприклад, методи зменшення розмірності, такі як PCA або t-SNE, можуть бути заздалегідь застосовані до кластеризації, щоб зменшити кількість вимірювань і, отже, обчислювальне навантаження. Крім того, наближені методи обчислення попарних відстаней, такі як хешування з урахуванням місцевості, можуть допомогти прискорити процес.

Агломеративна кластеризація знаходить застосування для текстової кластеризації в організації документів, тематичному моделюванні та резюмуванні. Наприклад, у дослідженнях його можна використовувати для кластеризації академічних статей щодо їхнього змісту, створюючи таким чином групи статей, які мають спільні напрямки дослідження або методології. Аналіз новин, наприклад, використовує агломеративну кластеризацію для групування статей за такими темами, як політика, спорт і технології, що може бути корисним для інформаційних агентств для організації великих обсягів вмісту. Подібним чином у сфері аналізу настроїв його можна використовувати для агломеративного групування тексту за аспектами позитивних, нейтральних або негативних відгуків.

Підводячи підсумок, можна сказати, що агломеративна кластеризація дійсно є сильним і універсальним алгоритмом кластеризації, який особливо підходить для завдань із текстовими даними. Побудова ієрархічних кластерів, які далі аналізуються на вищих або більш детальних рівнях, робить цей підхід дуже корисним для дослідження та структурування великих текстових корпусів. Хоча існують такі проблеми, як обчислювальна складність і чутливість до шуму, ретельна попередня обробка та зменшення розмірності можуть допомогти зменшити ці проблеми. Забезпечуючи гнучкий та інтуїтивно зрозумілий спосіб кластеризації, агломеративна кластеризація

залишається одним із важливих методів у машинному навчанні, який пропонує цінне розуміння структури текстових даних у широкому діапазоні програм.

2.4 Алгоритм LDA

Прихований розподіл Діріхле – це одна з найпопулярніших імовірнісних моделей, яка використовується для кластеризації тексту та пошуку тем, метод, згідно з яким алгоритм визначатиме певні моделі або теми, що існують у колекції документів. На основі генеративної ймовірнісної моделі документ розглядається як суміш тем. Кожна тема розподілена за словами. LDA є особливо потужним у виявленні прихованих тематичних структур у великих текстових корпусах, отже, робить його дуже важливим інструментом у НЛП та пошуку інформації [4, 7]. Потужність LDA полягає в тому, що він здатний моделювати базову структуру теми без мічених даних – неконтрольований метод кластеризації тексту. Інтуїція LDA полягає в тому, що документ у корпусі складається із суміші тем, де кожна тема є розподілом слів у словнику. Ці теми не є заздалегідь визначеними, а були виведені з даних, так що метою є досягнення представлення корпусу, де розкривається прихована структура тем. LDA передбачає фіксовану кількість тем, але на практиці це число зазвичай надається як вхідні дані користувачем або визначається за допомогою таких методів, як перехресна перевірка. Потім кожен документ у колекції розглядається як імовірнісна суміш цих тем, і LDA намагається знайти ці теми на основі розподілу слів у документах.

У своєму базовому налаштуванні LDA базується на трьох основних елементах (рис. 2.2): розподіл тем для кожного документа, розподіл слів для кожної теми та призначення тем для кожного слова в кожному документі. Вони поєднуються разом у генеративному процесі. По-перше, LDA передбачає, що кожен документ генерується випадковим процесом, де для

кожного слова в документі вибирається конкретна тема. Тоді кожен документ є сумішшю цих тем, причому кожна тема вносить певний внесок у документ. Ці тематичні пропорції взяті з розподілу Діріхле, розподілу ймовірностей, який часто використовують для моделювання розподілу за категоріальними змінними. Розподіл Діріхле особливо добре підходить для цієї мети, оскільки він гарантує, що сума ймовірностей тем для документа дорівнює одиниці. Кожна тема, у свою чергу, моделюється як розподіл слів, і передбачається, що ці розподіли слів незалежні один від одного [10, 11].

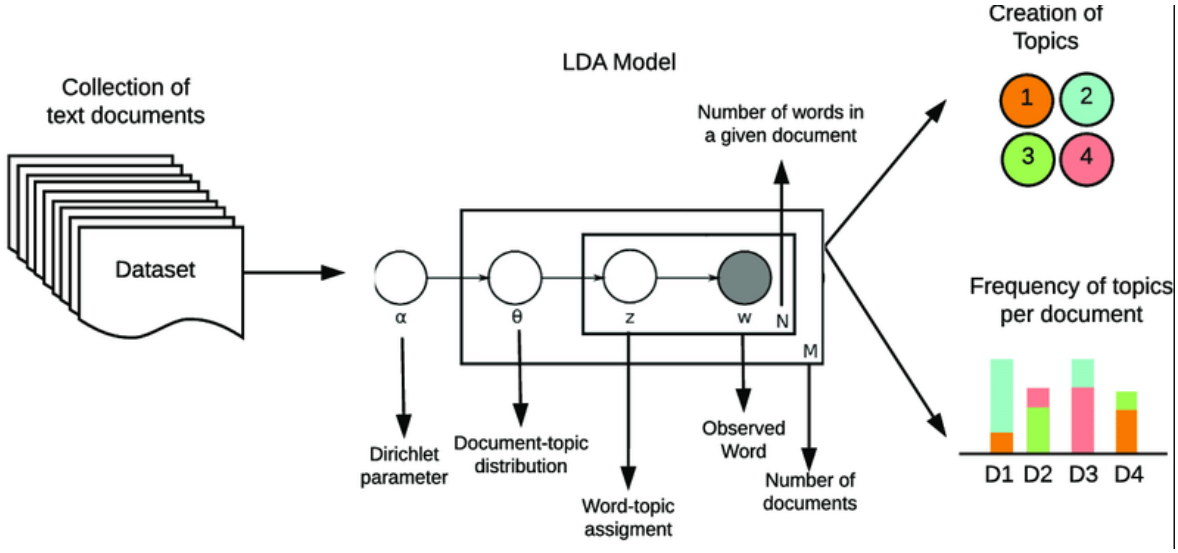

Рисунок 2.2 – Алгоритм LDA

У контексті кластеризації текстових даних LDA можна розуміти як процес, який намагається реконструювати генеративний механізм, за допомогою якого були створені документи. Маючи набір документів, LDA намагається зробити висновок про основні теми, які могли створити документи, і розподіл слів у кожній темі. Алгоритм робить це шляхом ітерації по корпусу, оновлюючи призначення тем для кожного слова в кожному документі на основі поточних оцінок розподілу тем і розподілу слів. Він шукає набір тем, які максимізують вірогідність спостережуваних даних, тобто слів і документів у корпусі.

Модель LDA визначається набором ключових параметрів, які описують генеративний процес. До них належать кількість тем (позначається як K), пріоріт Діріхле для розподілу тем (α), пріоріт Діріхле для розподілу слів (β) і призначення тем для кожного слова в документах. Пріори Діріхле контролюють розрідженість розподілу тем і слів, впливаючи на те, наскільки концентрованими чи розсіяними є теми та слова. Високе значення α заохочує документи до більш рівномірного розподілу за темами, кожен документ, імовірно, міститиме суміш багатьох тем, тоді як низьке значення α заохочує документи зосереджуватися на меншій кількості тем. Подібний ефект спостерігається для тем з точки зору значення для β, яке контролює розрідженість розподілу слів у темі [11]. Велике β дозволяє кожній темі містити більшість слів із однаковою ймовірністю, а менше β призводить до дуже гострих тем, сильно зосереджених навколо кількох слів. Цей аналіз показує, що LDA в основному спирається на процедуру виведення для оцінки параметрів моделі; зазвичай це досягається за допомогою таких алгоритмів, як вибірка Гіббса або варіаційний висновок. Вибірка Гіббса – це тип MCMC, який генерує послідовність вибірок із розподілу, що цікавить, таким чином, щоб модель могла досліджувати простір параметрів і сходитися до набору значень, які максимізують вірогідність спостережуваних даних. У випадку LDA вибірка Гіббса – це ітеративний процес оновлення присвоєння тем кожному слову в кожному документі з урахуванням поточних оцінок розподілу тем і слів. Після достатньої кількості ітерацій алгоритм збігається до стабільного набору розподілу тем, розподілу слів і призначення тем, які використовуються для опису основної структури корпусу.

Алгоритм LDA ініціалізує призначення теми для кожного слова в корпусі випадковим чином. Він припускає, що кожен документ є пакетом слів, і для кожного слова в цьому документі він вибирає випадкову тему. Випадково ініціалізована модель служить відправною точкою для цього алгоритму, з якого призначення тем поступово уточнюються з часом, використовуючи дані.

Це передбачає ітераційне оновлення призначень тем шляхом розгляду для кожного слова двох факторів: ймовірність присвоєння теми слову з огляду на розподіл тем у документі. І ймовірність того, що слову буде присвоєно певну тему, враховуючи розподіл слів у темі. Потім ці дві ймовірності об'єднуються для обчислення ймовірності теми для кожного слова. Він оновлює призначення теми для кожного слова на основі цього [12].

На кожному кроці процесу вибірки Гіббса призначення теми слова оновлюється щодо поточних тем документа та розподілу слів тієї самої теми. Ключова концепція полягає в тому, що слово, яке з'являється в документі, з великою часткою ймовірності належить до широко представленої теми в документі та повинно мати високу ймовірність утворення саме цього слова. Якщо слово, скажімо, «кіт», дуже часто зустрічається в документі, а тема «Тварини» має високу ймовірність генерувати слово «кішка», тоді алгоритм, напевно, помістить слово «кішка» в Тема «Тварини». Він оновлює призначення тем для всіх слів, після чого оновлює розподіл тем і слів, використовуючи нещодавно оновлені призначення тем.

Таким чином, оновлення тем і їх розподілів є ітеративним процесом, який постійно конвергується, щоб призначення тем для кожного слова стабілізувалися, а результат алгоритму забезпечує конвергентні розподіли тем для кожного документа та розподіл слів для кожної теми. Тематичний розподіл документа – це частка кожної теми в документі, а розподіл слів у темі – це ймовірність того, що кожне слово буде пов'язане з темою.

На практиці LDA в основному використовується для моделювання тем, де мета полягає в тому, щоб розкрити основні теми або теми в колекції документів. Наприклад, LDA може витягти теми «політика», «економіка», «спорт» і «технології» з великого корпусу новинних статей. Кожна стаття буде представлена як розподіл за такими темами, і алгоритм призначатиме більшу частину таким темам, як «політика» чи «спорт», у випадках, коли стаття обговорює будь-яку тему більше. Крім того, LDA дозволяє кластеризувати

документи, об’єднуючи їх на основі розподілу тем. Порівняння тематичного розподілу різних документів дасть змогу визначити, які документи подібні один до одного щодо їхнього змісту, таким чином уможливлюючи організацію великих колекцій тексту.

Тим не менш, LDA має певні обмеження, незважаючи на свої сильні сторони. Однією з головних проблем є те, що він припускає, що теми є незалежними, що не завжди може бути у реальних даних. У деяких випадках теми можуть бути дуже корельованими, і, отже, припущення незалежності може обмежити здатність моделі правдиво описувати дані. Іншим обмеженням LDA є те, що алгоритм вимагає попередньої специфікації кількості бажаних тем і оптимального отримання цього [13]. Методи можуть допомогти зробити це, наприклад, використовуючи такі методи, як перехресна перевірка, де використовуються спеціальні показники.

Крім того, LDA також може бути чутливим щодо якості даних і етапів їх попередньої обробки. Наприклад, LDA значною мірою покладається на дані про частоту термінів, і якщо ці дані стають надто шумними (можливо, через стоп-слова чи нерелевантні терміни), створені теми можуть не мати особливого сенсу. Знову ж таки, для покращення якості моделі потрібна певна попередня обробка тексту, така як видалення стоп-слова, корінь або лемматизація. Крім того, LDA може вийти з ладу для дуже великих або розріджених наборів даних, оскільки обчислювальна складність може стати досить високою. Однак такі методи, як стохастичний варіаційний висновок, були розроблені, щоб зробити LDA більш масштабованим для великих наборів даних.

Нарешті, прихований розподіл Діріхле є сильною імовірнісною моделлю для кластеризації текстових даних і розкриття основних тем у колекції документів. Гнучкість, здатність знаходити приховані структури та неконтрольований характер надають йому високу цінність у багатьох програмах, від кластеризації документів до тематичного моделювання в таких

галузях, як цифрові гуманітарні науки, пошук інформації та аналіз соціальних мереж. Незважаючи на свої обмеження, LDA залишається одним із найпопулярніших і найефективніших методів виявлення прихованої семантичної структури в тексті, що дозволяє дослідникам і практикам упорядковувати, аналізувати та отримувати цінну інформацію з великих колекцій неструктурованих текстових даних.

2.5 Алгоритми на основі нейронних мереж для кластеризації текстів

В основі методів кластеризації, заснованих на нейронних мережах, лежить сила впроваджень навчання або, іншими словами, низьковимірних векторних представлень тексту. Ці вбудовування фіксують семантичну інформацію про текст таким чином, що алгоритм кластеризації працюватиме в просторі, де відстані між точками або документами відображають семантичну, а не синтаксичну подібність [15]. Це величезний відхід від традиційних підходів, які зазвичай покладалися на прямі представлення на основі частоти термінів або евристичних методів для визначення подібності документів. Навчаючись на даних, нейронні мережі можуть виділяти тонкі зв’язки між словами, фразами та документами, що покращує якість процесу кластеризації.

Чітке розуміння того, як нейронні мережі використовувалися для кластеризації тексту, вимагає перегляду основної проблеми в традиційних термінах. Кластеризація тексту означає процес побудови груп або кластерів таким чином, що документи з одного кластера набагато більше схожі один на одного, ніж будь-який документ в іншому кластері. У деяких простих моделях документи розглядаються як вектори в деякому багатовимірному просторі, де кожен конкретний вимір відповідає певному унікальному словнику словника. Це представлення зазвичай розріджене, особливо коли мова йде про великі корпуси, оскільки кожен документ містить лише невелику підмножину всіх

можливих слів. Традиційні алгоритми кластеризації, такі як k-середні, працюють на таких векторах великої розмірності. Однак через велику розмірність їх продуктивність зазвичай погіршується через розрідженість векторів і нездатність традиційних показників відстані охоплювати складні зв’язки між словом і документом, такі як евклідова відстань.

Методи, засновані на нейронних мережах, намагаються вирішити це, представляючи вхідний текст у формі щільних, низькорозмірних вставок, фіксуючи семантичне значення тексту. Вони отримуються в результаті навчання нейронних мереж, які представляють заданий вхідний текст таким чином, що схожі тексти відображаються на точки, розташовані близько в просторі, а несхожі тексти – у точки, що лежать далі. Word2Vec – це одна з найпопулярніших моделей нейронних мереж для вивчення вбудовування слів шляхом навчання зіставлення слів у щільні векторні представлення на основі їхнього контекстного використання у великому корпусі [12]. Після того, як слова вбудовано у вектори, документи можна представити як сукупність вбудованих слів, які вони містять. Але така попередня обробка перетворює вхідні дані для кластеризації в простір, де схожість дійсно стоїть за схожістю, а не за частотами. Загалом, word2vec навчається або за допомогою однієї з двох можливих архітектур, або за допомогою деякого їх об’єднання, а саме: CBOW і Skip-gram. Модель CBOW передбачає цільове слово за допомогою контекстного вікна навколишніх слів, тоді як модель Skip-gram робить навпаки, прогнозуючи навколишні слова за цільовим словом. Ключова ідея цих архітектур полягає в тому, що слова, які з’являються в подібних контекстах, мають значення, які також схожі. Таким чином, це призводить до того, що вкладення знаходяться ближче у векторному просторі. Здатність Word2Vec створювати щільні вбудовування слів – це те, що робить його таким потужним інструментом для завдань текстової кластеризації: тепер документи можна представляти як вектори, які краще фіксують їхній базовий семантичний вміст.

Окрім Word2Vec, інші передові моделі нейронних мереж, такі як GloVe (GloVe (Global Vectors for Word Representation)) і нещодавні моделі на основі трансформаторів, включаючи BERT (Bidirectional Encoder Representations from Transformers), розвинули знання про техніку вбудовування тексту далі. У той час як Word2Vec і GloVe вивчають представлення на основі статистики спільного входження, BERT вивчає контекстуалізовані вбудовування слів за допомогою архітектури трансформатора – представлення слова змінюватиметься відповідно до контексту. Захоплення контекстних нюансів є важливою сильною стороною BERT, і з цієї причини він найкраще підходить для кластеризації тексту, оскільки можна досягти набагато кращого та значущого представлення документів [9]. Ці контекстні вбудовування формують основу для алгоритмів кластеризації, які дозволяють згрупувати схожі значення документів, навіть якщо використовується різна лексика чи формулювання.

Коли вставлення слова або документа стане доступним, можна використовувати кілька алгоритмів кластеризації, щоб кластеризувати подібні тексти разом. Як альтернатива, нейронні мережі можна безпосередньо використовувати для кластеризації за допомогою спеціалізованих моделей, які вивчають разом як вбудовування, так і кластери. Ці моделі часто покладаються на автокодери або нейронні мережі, навчені стискати перед реконструкцією вхідних даних. Автокодер складається з мережі кодувальника, яка проектує вхідні дані в представлення нижчої розмірності, тоді як мережа декодера реконструює вхідні дані з цього стисненого представлення. Під час навчання автокодувальника для мінімізації помилок у реконструкції мережа вивчає компактне представлення даних, у якому фіксуються найбільш помітні характеристики. Потім вихідні дані кодера можна використовувати як вбудовування вхідного тексту.

Дуже ефективний метод кластеризації з використанням нейронних мереж навчає глибокий автокодувальник, у якому неконтрольовані цілі

кластеризації поєднуються з можливістю автокодувальника вивчати значущі вбудовування. Ці моделі зазвичай називаються Deep Embedded Clustering (DEC), а їхня процедура навчання оптимізує представлення даних у просторі вбудовування разом із призначенням точок даних кластерам. По-перше, модель DEC попередньо навчає автокодувальник завданням вивчення компактного представлення, а потім використовує алгоритми кластеризації, такі як k-середні, для призначення точок даних [13]. Глибока нейронна мережа навчається шляхом мінімізації як помилки реконструкції, так і втрат кластеризації, таким чином навчені вбудовування є компактними та дискримінаційними, що забезпечує кращу кластеризацію. Цей підхід поєднує в собі сильні сторони нейронних мереж у навчанні функцій із звичайними алгоритмами кластеризації для забезпечення високоякісних та інтерпретованих кластерів.

Іншим основним методом, який включає нейронні мережі в процес кластеризації, є Самоорганізуючі карти (SOM). SOM – це свого роду алгоритм неконтрольованого навчання, який використовує нейронні мережі для відображення високовимірних даних на деяку низьковимірну сітку, зберігаючи при цьому топологічну структуру даних. Найкорисніші для текстової кластеризації, SOM зазвичай застосовуються, коли потрібна інтерпретація, графічно показуючи, як різні документи потрапляють у різні кластери. Вагові коефіцієнти в сітці коригуються завдяки вхідній нейронній мережі, яка зрештою починає організовувати сітку так, щоб подібні точки даних розташовувалися найближче одна до одної. У контексті текстової кластеризації текстові дані спочатку будуть підготовлені як вбудовування, а потім за допомогою алгоритму SOM їх можна буде впорядкувати в кластери семантичної подібності.

Не обов’язково методи нейронної мережі для кластеризації тексту мають базуватися на вбудовуванні. Методології CNN і RNN також досліджувалися для завдань текстової кластеризації, але зазвичай вони зосереджені на

класифікації окремих документів або створенні вставок для подальших завдань у конвеєрі. CNN є потужними у моделюванні локальних шаблонів у тексті. Наприклад, певна комбінація слів або n-грам може вказувати на приналежність документа до одного кластера порівняно з іншим. Мережі RNN, а особливо ті, що включають мережі LSTM, мають невід'ємну здатність вивчати послідовні залежності в текстах, що робить їх дуже корисними для кластеризації документів, де важливий порядок слів/речень. Хоча CNN і RNN не використовуються безпосередньо для кластеризації, вони можуть бути включені в конвеєр кластеризації шляхом створення вставок, які потім можуть бути кластеризовані за допомогою традиційних алгоритмів кластеризації або методів кластеризації на основі нейронної мережі.

Ключова перевага використання нейронних мереж для кластеризації тексту полягає в тому, що вони можуть вивчати складні високорівневі функції з необроблених даних без потреби в ручних функціях. На відміну від традиційних методів кластеризації, які спираються на прості показники відстані або статистичні припущення щодо даних, нейронні мережі вивчають уявлення, глибоко поінформовані самими даними [16]. Це призведе до вбудовування, яке є більш репрезентативним для семантичних зв'язків між документами, таким чином забезпечуючи більш точні та інтерпретовані кластери. Крім того, нейронні мережі мають високу адаптивність, дозволяючи включати передові методи, такі як механізми уваги, трансформатори та навчання з підкріпленням, які ще більше покращують продуктивність кластеризації.

Однак є деякі проблеми, пов'язані з методами кластеризації на основі нейронної мережі. Одна з головних проблем полягає в тому, що для навчання потрібно багато мічених даних, особливо у випадку контрольованих методів. Неконтрольовані методи, автокодер і глибока вбудована кластеризація не використовують жодних мічених даних, але часто потребують значних обчислювальних ресурсів і великих наборів даних для належної роботи. Крім

того, ці глибокі нейронні мережі недешеві і можуть вимагати використання спеціального апаратного забезпечення, наприклад графічних процесорів. У той час як підготовка та навчання за такими великими нейронними мережами пов’язані з кількома труднощами, переваги, які вони приносять у вигляді гнучкості, точності та здатності фіксувати складні семантичні зв’язки між текстами, роблять їх дуже потужними для виконання завдань текстової аналітики.

Зрештою, алгоритми кластеризації тексту на основі нейронної мережі представляють собою серйозний крок за межі більш традиційних підходів до кластеризації, особливо в їхній здатності вивчати багаті семантичні представлення текстових даних. У додатках, що включають вбудовування слів, автоматичне кодування, глибоку вбудовану кластеризацію та самоорганізовані карти, серед інших методологій, нейронні мережі цілком здатні запропонувати гнучкі, високоефективні рішення для проблем кластеризації тексту [1, 5]. Ці методи особливо корисні в контекстах, де традиційні методи кластеризації мають труднощі, як-от великорозмірні, неструктуровані текстові дані. Хоча залишаються проблеми щодо витрат на обчислення та вимог до даних, продовження розробки моделей нейронних мереж для кластеризації тексту обіцяє подальше підвищення їх застосовності та ефективності в широкому діапазоні областей, від пошуку інформації та організації документів до тематичного моделювання та аналізу настроїв.

2.6 Висновки до другого розділу

Дослідження методів кластеризації для текстових даних підкреслило різноманітність підходів, які можна застосувати для вилучення значимих шаблонів із великих неструктурованих наборів даних. Кластеризація тексту є важливою частиною обробки природної мови, що дозволяє класифікувати та організовувати великі корпуси текстів у групи подібного змісту. Традиційні

методи кластеризації тексту, такі як k-середні, ієрархічна кластеризація та прихований розподіл Діріхле, усі працюють на основі досить простих математичних передумов і припущень щодо даних. Ці методи були дуже успішними в широкому діапазоні застосувань, особливо коли структура даних є чіткою та добре розділеною. Однак їхні недоліки, головним чином чутливість до розрідженості ознак, залежність від фіксованих припущень щодо розподілу тем і нездатність охопити складні семантичні зв'язки, викликали необхідність у більш складних техніках.

Зі зростанням розвитку НЛП поява моделей на основі нейронних мереж значно змінила ландшафт кластеризації тексту. Незважаючи на те, що нейронні мережі можуть навчатися на основі тонкощів даних, вони запровадили нові рівні гнучкості та складності для завдань кластеризації. Ці моделі генерують низькорозмірні щільні вбудовування текстових даних, які інкапсулюють семантичне значення документів за допомогою архітектур глибокого навчання.

Хоча методи кластеризації на основі нейронної мережі мають значні переваги, впровадження також супроводжує проблеми. Існує потреба в значних обчислювальних ресурсах, тому що навчання глибоких нейронних мереж є обчислювально дорогим і трудомістким. Крім того, багато позначених даних все ще потрібно для навчання, що стає важливим обмеженням для багатьох програм, переважно під час виконання під наглядом. Навіть у неконтрольованих налаштуваннях обсяг даних, необхідний для ефективного навчання нейронних мереж, може бути складним для підтримки як обчислень, так і доступності даних. Крім того, незважаючи на те, що нейронні мережі надзвичайно добре вивчають складні уявлення, інтерпретація цих моделей все ще недостатня. Зрозуміло, що за своєю природою дуже важко зрозуміти міркування, що стоять за рішенням цих моделей кластеризуватися. Отже, це обмежує їх застосування в областях, де прозорість і пояснюваність дійсно необхідні.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Постановка задачі та функціональні вимоги

Необхідно розробити програму для аналізу текстових документів із використанням методів кластеризації.

Основні функціональні вимоги до програми:

– виконання очищення тексту: видалення зайвих символів, приведення до нижнього регістру;

– токенізація тексту та виключення стоп-слів на різних мовах;

– побудова векторного представлення текстових даних з використанням TF-IDF (з можливістю обмеження кількості ознак);

– реалізація декількох методів кластеризації (наприклад, k-means та ієрархічна кластеризація);

– автоматичний вибір кількості кластерів залежно від розміру вхідних даних;

– розрахунок метрик для кожного методу;

– порівняння результатів кластеризації на основі отриманих метрик;

– побудова графіків для аналізу розподілу документів між кластерами;

– виконання зменшення розмірності даних до 2D-простору для візуального представлення кластерів;

– генерація теплових карт для відображення відстаней між кластерами;

– формування текстового звіту, що містить статистику розподілу документів по кластерах, ключові метрики, а також приклади текстів у кожному кластері.

Програма має бути розроблена таким чином, щоб забезпечити гнучкість та модульність, дозволяючи легко розширювати її функціонал для інших задач аналізу тексту.

3.2 Використані технології та інструменти

Розробка програмного забезпечення, орієнтованого на аналіз текстових даних, включала використання сучасних технологій та інструментів, які забезпечують ефективність, продуктивність та гнучкість реалізації. Ключовими компонентами, що формують основу технологічного стеку, є бібліотеки для обробки природної мови, алгоритми машинного навчання, інструменти для візуалізації даних та засоби для роботи з різноманітними форматами вхідних і вихідних даних.

Обробка природної мови здійснювалася за допомогою бібліотеки NLTK (Natural Language Toolkit), яка є однією з найпоширеніших і найбільш розвинених платформ для роботи з текстовими даними. Використання NLTK дозволило реалізувати функції токенізації, видалення стоп-слів, стемінгу та лематизації, які є критичними для підготовки тексту до аналізу. Токенізація, виконана за допомогою word_tokenize, дозволила розбити текстові дані на слова, зберігаючи контекст розділових знаків і складних конструкцій. Завдяки підтримці багатьох мов, бібліотека забезпечила можливість ефективної обробки текстів різними мовами, включаючи англійську та українську. Лематизація здійснювалася з використанням класу WordNetLemmatizer, що базується на WordNet – великій лексичній базі, яка забезпечує розпізнавання лем відповідно до частини мови, враховуючи контекст.

Для обчислення числових характеристик текстів використовувалася бібліотека Scikit-learn, зокрема її модуль TfidfVectorizer. Цей інструмент дозволяє перетворити текстові документи на числові вектори, враховуючи частотність термінів у документі та їхню значущість у межах корпусу. Такий підхід забезпечив збереження найбільш важливих характеристик тексту для подальшого аналізу. Scikit-learn також був використаний для реалізації алгоритмів кластеризації. Метод K-Means забезпечував розподіл документів між кластерами на основі мінімізації внутрішньогрупової дисперсії. Для

виконання ієрархічної кластеризації використовувався клас AgglomerativeClustering, який дозволив створювати ієрархічну структуру кластерів, що є важливим для аналізу текстових даних у контексті їхньої схожості.

Оцінка результатів кластеризації здійснювалася за допомогою метрик, доступних у бібліотеці Scikit-learn, таких як silhouette_score, calinski_harabasz_score і davies_bouldin_score. Вибір цих метрик був обумовлений необхідністю забезпечення багатогранного аналізу якості кластеризації. Наприклад, silhouette_score дозволяє оцінити щільність кластерів і їхню відокремленість, тоді як calinski_harabasz_score аналізує співвідношення міждисперсійної та внутрішньодисперсійної складових. Метрика davies_bouldin_score, у свою чергу, мінімізує внутрішньогрупові відхилення відносно міжгрупових, що дозволяє точно оцінити внутрішню структуру кластерів.

Засоби візуалізації даних у проєкті базуються на бібліотеках Matplotlib і Seaborn. Matplotlib використовувалася для створення базових графіків, таких як розподіл кластерів, динаміка метрик якості кластеризації та 2D-проєкції зменшеної вимірності. Використання класу pyplot забезпечило гнучкість і простоту у налаштуванні візуальних параметрів графіків, таких як розміри, колірні схеми та підписи осей. Seaborn, як інструмент для створення більш естетичних і деталізованих візуалізацій, забезпечив побудову теплових карт відстаней між кластерами та розподілу слів у межах кластерів. У поєднанні ці дві бібліотеки створили можливість для генерації високоякісних графічних зображень, які сприяють кращому розумінню результатів аналізу.

Крім того, в проєкті використовувалися інструменти для зменшення розмірності даних. Метод головних компонент (PCA) із бібліотеки Scikit-learn дозволив виконати проєкцію багатовимірних даних у двовимірний простір для візуалізації кластерів. Завдяки цьому стало можливим аналізувати результати кластеризації навіть у випадках, коли початкові дані мали високу розмірність.

У процесі роботи з даними також застосовувалися бібліотеки NumPy і Pandas. NumPy забезпечувала ефективне виконання операцій із багатовимірними масивами, зокрема при обчисленні відстаней між текстовими векторами. Pandas використовувалася для зручного управління текстовими даними у форматі таблиць, що спрощувало маніпуляції з вхідними та вихідними даними. Крім того, можливості Pandas дозволили автоматизувати створення підсумкових звітів, які включають результати кластеризації та аналізу якості.

Для забезпечення багатомовності у модулі обробки тексту реалізована інтеграція зі словниками мовних ресурсів, доступних через бібліотеки NLTK та додаткові зовнішні джерела. Це дозволило підвищити точність обробки текстів різними мовами та врахувати специфіку синтаксису і морфології.

Інтеграція описаних технологій і засобів у рамках єдиної архітектури забезпечила створення інструменту, здатного працювати з великими наборами текстових даних, надаючи користувачам глибокий аналіз та зручний спосіб візуалізації результатів. Кожен компонент технологічного стеку був обраний і налаштований таким чином, щоб забезпечити максимальну продуктивність, масштабованість і адаптивність програмного забезпечення до різноманітних завдань у сфері аналізу тексту.

3.3 Структура та архітектура програмного забезпечення

Архітектура розробленого програмного коду побудована на основі принципів модульності, гнучкості та розширюваності. Основна концепція проєкту спрямована на створення інструменту для глибокого аналізу текстових даних із використанням сучасних методів машинного навчання та кластеризації. У центрі архітектури знаходиться клас AdvancedTextClusterAnalyzer (рис. 3.1), який забезпечує виконання всіх ключових етапів обробки тексту, кластеризації та візуалізації результатів.

Його структура відображає логічний поділ завдань на окремі підпроцеси, що сприяє підтримуваності та масштабованості програмного забезпечення.

```
class AdvancedTextClusterAnalyzer:
    def __init__(self, documents):
        """
        Розширена ініціалізація аналізатора кластеризації текстових документів.
        """
        # Фільтрація порожніх документів
        self.documents = [doc for doc in documents if doc and isinstance(doc, str) and len(doc.strip()) > 0]

        if len(self.documents) == 0:
            raise ValueError("Немає коректних текстових документів для аналізу!")

        self.preprocessed_docs = []
        self.vectorizer = None
        self.feature_matrix = None
        self.clustering_results = {}

    def preprocess_text(self, text):
        """
        Поглиблена попередня обробка тексту.
        """
        text = text.lower()
        text = re.sub(r'[^а-яа-z\s]', '', text)
        tokens = word_tokenize(text)
        stop_words = set(stopwords.words('russian') + stopwords.words('english'))
        tokens = [token for token in tokens if token not in stop_words]
        return ' '.join(tokens)

    def prepare_documents(self):
        """
        Підготовка та векторизація документів.
        """
        self.preprocessed_docs = [self.preprocess_text(doc) for doc in self.documents]

        if len(self.preprocessed_docs) == 0:
            raise ValueError("Після попередньої обробки не залишилось документів!")

        self.vectorizer = TfidfVectorizer(max_features=1000)
        self.feature_matrix = self.vectorizer.fit_transform(self.preprocessed_docs)
```

Рисунок 3.1 – Програмний код класу AdvancedTextClusterAnalyzer

На етапі ініціалізації класу обробляються вхідні дані. Архітектура реалізує механізм перевірки введених даних, що виконується для забезпечення коректності роботи програми з текстами. Зокрема, перевіряється, чи є кожен із документів текстовим рядком, і чи вони не є порожніми. Цей підхід дозволяє уникнути помилок під час виконання подальших етапів обробки. Якщо вхідні дані не відповідають визначеним критеріям, програма генерує виняток із детальним описом помилки. Це рішення базується на принципі раннього

виявлення проблем, що є критично важливим для забезпечення стабільності програмного забезпечення в умовах роботи з великими обсягами текстових даних.

Попередня обробка тексту є одним із найважливіших етапів у контексті задач кластеризації. Для її виконання в класі реалізована методика очищення текстів, яка передбачає приведення тексту до нижнього регістру, видалення небажаних символів, таких як розділові знаки та спеціальні символи, а також фільтрацію стоп-слів. Особливістю цієї реалізації є те, що вона враховує багатомовність текстів, використовуючи вбудовані ресурси бібліотеки NLTK. Це дозволяє працювати зі стоп-словами як англійською, так і українською мовами. Сегментація тексту на токени забезпечується за допомогою методу word_tokenize (рис. 3.1), що дозволяє зберігати точність розбиття навіть у випадках, коли текст містить складні синтаксичні конструкції. Завершальним кроком у цьому процесі є об’єднання токенів у текст, підготовлений для подальшої обробки.

Архітектура класу включає етап векторизації тексту, який реалізується за допомогою методу TfidfVectorizer. Ця техніка вибрана через її здатність виділяти значущі ознаки текстів, які відображають важливість слів у межах документа та корпусу загалом. Векторизація обмежується тисячею ознак, що оптимізує пам’ять і підвищує швидкість обробки без значної втрати точності кластеризації. Створення матриці ознак супроводжується валідацією, яка дозволяє перевірити, чи створена матриця містить достатню кількість даних для подальшого аналізу. У разі виникнення помилок, пов’язаних із браком інформації, генерується відповідне повідомлення для користувача.

Ключовим компонентом архітектури є модуль кластеризації. Він підтримує кілька методів, таких як K-Means та ієрархічна кластеризація. Архітектура забезпечує динамічний вибір кількості кластерів, що залежить від розміру вхідних даних. Це дозволяє адаптувати алгоритми до різноманітних наборів даних без необхідності ручного налаштування параметрів. Метод K-

Means реалізований із використанням бібліотеки Scikit-learn. Він забезпечує створення кластерів шляхом розподілу текстів відповідно до їхніх векторних представлень. Для оцінки якості кластеризації використовуються метрики silhouette_score, calinski_harabasz_score і davies_bouldin_score. Ці метрики дозволяють визначити щільність кластерів, їхню відокремленість та ступінь внутрішньої неоднорідності.

Ієрархічна кластеризація, яка також є частиною архітектури, реалізована через AgglomerativeClustering. Вона дозволяє досліджувати текстові дані з точки зору їхньої ієрархічної структури. Архітектура програми включає механізм попередньої обробки матриці ознак, який забезпечує перетворення розріджених матриць у щільний формат. Це необхідно для забезпечення коректної роботи ієрархічних алгоритмів кластеризації. Оцінка результатів виконується аналогічно K-Means, що дозволяє порівнювати методи між собою.

Особливу увагу приділено реалізації модуля візуалізації. Цей компонент забезпечує створення графічних представлень, які допомагають аналізувати результати кластеризації. Він включає побудову порівняльних графіків метрик кластеризації, розподілу документів між кластерами, 2D-проєкцій зменшеної вимірності, теплових карт відстаней між кластерами та розподілу слів у межах кластерів. Візуалізація виконується з використанням бібліотек Matplotlib і Seaborn, які дозволяють створювати високоякісні графіки з широкими можливостями налаштування. Модуль також включає засоби інтерактивного порівняння методів кластеризації, що є корисним для прийняття рішень під час роботи з великими наборами текстових даних.

Завершальним компонентом архітектури є модуль генерації текстового звіту. Цей модуль надає користувачеві детальну інформацію про результати кластеризації, включаючи розподіл документів між кластерами, значення метрик і приклади текстів у кожному кластері. Реалізація базується на

принципах прозорості та читабельності, що дозволяє отримати повне уявлення про процес кластеризації та її результати.

Архітектура програмного забезпечення забезпечує масштабованість завдяки модульному дизайну, який дозволяє легко додавати нові методи обробки тексту, алгоритми кластеризації або метрики оцінки. Гнучкість реалізації дозволяє адаптувати програму для вирішення специфічних завдань, пов'язаних із аналізом тексту. Крім того, використання сучасних бібліотек, таких як NLTK, Scikit-learn і Matplotlib, гарантує високу продуктивність і надійність програмного коду. Програма також має механізми обробки винятків, що забезпечує її стабільність у випадках некоректних вхідних даних або інших помилок.

На рисунку 3.2 наведено реалізацію кластеризації методом K-means з розширеною аналітикою, а на рисунку 3.3 – ієрархічну кластеризацію з розширеною аналітикою.

```
def _kmeans_clustering(self, n_clusters=None):
    """
    Кластеризація методом K-means з розширеною аналітикою.
    """
    if n_clusters is None:
        n_clusters = min(5, max(2, len(self.documents)//2))

    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    labels = kmeans.fit_predict(self.feature_matrix)

    return {
        'labels': labels,
        'centroids': kmeans.cluster_centers_,
        'silhouette_score': silhouette_score(self.feature_matrix, labels),
        'calinski_harabasz_score': calinski_harabasz_score(self.feature_matrix.toarray(), labels),
        'davies_bouldin_score': davies_bouldin_score(self.feature_matrix.toarray(), labels)
    }
```

Рисунок 3.2 – Реалізація кластеризації методом K-means

```
def _hierarchical_clustering(self, n_clusters=None):
    """
    Ієрархічна кластеризація з розширеною аналітикою.
    """
    if n_clusters is None:
        n_clusters = min(5, max(2, len(self.documents)//2))

    hierarchical = AgglomerativeClustering(n_clusters=n_clusters)
    labels = hierarchical.fit_predict(self.feature_matrix.toarray())

    return {
        'labels': labels,
        'silhouette_score': silhouette_score(self.feature_matrix, labels),
        'calinski_harabasz_score': calinski_harabasz_score(self.feature_matrix.toarray(), labels),
        'davies_bouldin_score': davies_bouldin_score(self.feature_matrix.toarray(), labels)
    }
```

Рисунок 3.3 – Реалізація ієрархічної кластеризації

Узагальнюючи, розроблений програмний код є потужним інструментом для аналізу текстових даних, який поєднує в собі сучасні алгоритми кластеризації, розвинену аналітику та інтуїтивно зрозумілу візуалізацію. Його архітектура побудована таким чином, щоб забезпечити простоту використання, масштабованість і адаптивність, що робить його придатним для широкого спектра завдань у сфері обробки природної мови та аналізу даних.

3.4 Опис роботи кластеризації

Процес роботи розробленого програмного забезпечення для аналізу текстових даних розпочинається із завантаження вхідних даних. Програма підтримує декілька форматів файлів, включаючи текстові файли, електронні таблиці та інші популярні формати, що забезпечує універсальність її застосування. Дані, що надходять на вхід, проходять початковий етап валідації. Це дозволяє переконатися, що всі необхідні поля присутні, а також перевірити коректність структури та форматування тексту. У разі виявлення невідповідностей програма надає зворотний зв’язок, повідомляючи про помилки або пропонуючи внести необхідні корективи.

Після успішного завершення валідації дані передаються до етапу попередньої обробки, який є одним із найважливіших у всьому процесі. На цьому етапі виконується очищення тексту, що включає видалення стоп-слів, спеціальних символів, чисел, а також зайвих пробілів. Для цього використовуються алгоритми регулярних виразів і бібліотеки обробки тексту, такі як NLTK. Додатково здійснюється нормалізація тексту, яка полягає у приведенні всіх символів до нижнього регістру. Це дозволяє уникнути дублювання інформації, пов’язаної із чутливістю до регістру. Лематизація та стемінг забезпечують приведення слів до базової форми, зберігаючи при цьому їхній семантичний зміст.

На наступному етапі здійснюється перетворення текстових даних у числовий формат. Це завдання вирішується за допомогою методів векторизації, серед яких найбільш ефективним є TF-IDF (term frequency-inverse document frequency). Цей метод дозволяє врахувати частотність появи слів у кожному документі, а також їхню значущість у загальному корпусі текстів. В результаті кожен документ перетворюється на вектор, що містить числові характеристики, придатні для подальшої обробки. Залежно від розміру корпусу текстів і обраних параметрів векторизації, можуть бути сформовані як розріджені, так і щільні матриці.

Після завершення етапу векторизації текстові дані стають доступними для алгоритмів кластеризації. Програма передбачає декілька підходів до кластеризації, серед яких найчастіше застосовуються K-Means та ієрархічна кластеризація. Алгоритм K-Means працює шляхом ітеративного визначення центрів кластерів та їхнього перегрупування відповідно до мінімізації внутрішньогрупової дисперсії. Початково центри кластерів вибираються випадковим чином, після чого виконується процес рекурсивної оптимізації, що завершується, коли зміни центрів стають незначними або досягається максимальна кількість ітерацій. Ієрархічна кластеризація, з іншого боку, формує дерево кластерів, починаючи з об’єднання найближчих точок і

поступово створюючи більші групи. Цей підхід забезпечує додаткову гнучкість в аналізі структури даних, дозволяючи обрати оптимальний рівень деталізації кластерів.

Результати кластеризації підлягають ретельній оцінці якості. Для цього використовуються метрики, які враховують внутрішню щільність кластерів, відстань між ними та загальну структуру даних. Наприклад, коефіцієнт Силуету вимірює, наскільки добре кожна точка вписується у свій кластер порівняно із сусіднім. Метрика Калінські-Харабаша аналізує співвідношення між міжкластерною та внутрішньокластерною дисперсіями, тоді як коефіцієнт Девіса-Булдіна мінімізує внутрішньогрупові відхилення відносно міжгрупових. Коли оцінка якості завершена, результати кластеризації передаються до модулів візуалізації. Програма використовує потужні засоби для створення графіків, що дозволяють відобразити структуру кластерів, розподіл точок у просторі та взаємозв’язки між ними. Наприклад, використання PCA (аналіз головних компонент) дозволяє зменшити розмірність даних до двох або трьох вимірів, зберігаючи при цьому основну інформацію. Отримані результати візуалізуються у вигляді розсіювання точок, теплових карт чи дендрограм, залежно від обраного алгоритму кластеризації.

Програма також підтримує можливість збереження результатів у різних форматах, таких як CSV або JSON. Це забезпечує зручність інтеграції отриманих даних з іншими аналітичними системами або подальшого використання у наукових дослідженнях. Крім того, користувач має можливість налаштовувати параметри кластеризації, методів оцінки якості та візуалізації відповідно до своїх потреб, що робить програму надзвичайно гнучкою та адаптивною.

Однією з важливих особливостей програмного забезпечення є його здатність обробляти великі обсяги даних завдяки оптимізованій архітектурі. Використання багатопотокової обробки дозволяє розподілити обчислювальні завдання між кількома ядрами процесора, що суттєво скорочує час виконання.

Крім того, розробка була здійснена з урахуванням принципів модульності, що дозволяє легко інтегрувати нові алгоритми чи функціональні модулі без значних змін у кодовій базі.

На завершальному етапі роботи програми користувач отримує підсумковий звіт, який включає візуалізації, оцінки якості кластеризації та рекомендації щодо інтерпретації результатів. Цей звіт може бути збережений у зручному для користувача форматі або відправлений на електронну пошту. Завдяки цьому програмне забезпечення не лише спрощує виконання складних аналітичних завдань, але й забезпечує зручність і прозорість результатів для кінцевих користувачів.

3.5 Проведення кластеризації

Для проведення тестування кластеризації з метою групування текстових даних було використано два методи: KMeans та Hierarchical.

Отриманий розподіл документів по кластерах з використанням метода KMeans наведено на рисунку 3.4. Метрики якості кластеризації під час використання цього метода наведено на рисунку 3.5. Отримані результати з використанням цього метода наведено на рисунку 3.6. На ньому зазначені приклади документів у різних кластерах. При цьому початковий набір текстових документів наведено на рисунку 3.7.

```
Звіт про кластеризацію текстових документів:

Метод: KMeans

Розподіл документів по кластерах:
  Кластер 2: 4 документів (33.33%)
  Кластер 1: 3 документів (25.00%)
  Кластер 3: 2 документів (16.67%)
  Кластер 0: 1 документів (8.33%)
  Кластер 4: 2 документів (16.67%)
```

Рисунок 3.4 – Розподіл документів по кластерах для метода KMeans

```
Метрики якості кластеризації:
  • Silhouette Score: 0.0685
  • Calinski-Harabasz Score: 1.4666
  • Davies-Bouldin Score: 1.4441
```

Рисунок 3.5 – Метрики якості кластеризації для метода KMeans

```
Приклади документів у кластерах:
  Кластер 2:
    - Машинне навчання є важливою галуззю штучного інтелекту...
    - Технології штучного інтелекту розвиваються дуже швидко...
  Кластер 1:
    - Глибоке навчання використовує нейронні мережі для аналізу даних...
    - Нейронні мережі дозволяють комп'ютерам навчатися як люди...
  Кластер 3:
    - Штучний інтелект змінює сучасний світ технологій...
    - Штучний інтелект підтримує автоматизацію процесів...
  Кластер 0:
    - Алгоритми машинного навчання покращують точність прогнозування...
  Кластер 4:
    - Комп'ютерне бачення використовує глибоке навчання...
    - Комп'ютерне навчання революціонізує сучасні технології...
```

Рисунок 3.6 – Приклади документів у кластерах для метода KMeans

```
documents = [
    "Машинне навчання є важливою галуззю штучного інтелекту",
    "Глибоке навчання використовує нейронні мережі для аналізу даних",
    "Штучний інтелект змінює сучасний світ технологій",
    "Нейронні мережі дозволяють комп'ютерам навчатися як люди",
    "Алгоритми машинного навчання покращують точність прогнозування",
    "Комп'ютерне бачення використовує глибоке навчання",
    "Технології штучного інтелекту розвиваються дуже швидко",
    "Машинне навчання має багато практичних застосувань",
    "Data Science допомагає аналізувати великі обсяги інформації",
    "Нейронні мережі застосовуються в різних галузях",
    "Штучний інтелект підтримує автоматизацію процесів",
    "Комп'ютерне навчання революціонізує сучасні технології"
]
```

Рисунок 3.7 – Початковий набір документів для тестування кластеризації

Наступні експерименти були проведені з використанням метода Hierarchical. Розподіл документів по кластерах наведено на рисунку 3.8. Метрики якості кластеризації для вказаного метода наведено на рисунку 3.9. Результати кластеризації з використанням цього метода наведено на рисунку 3.10. Початковий набір документів для тестування кластеризації та проведення експериментів наведено на рисунку 3.7.

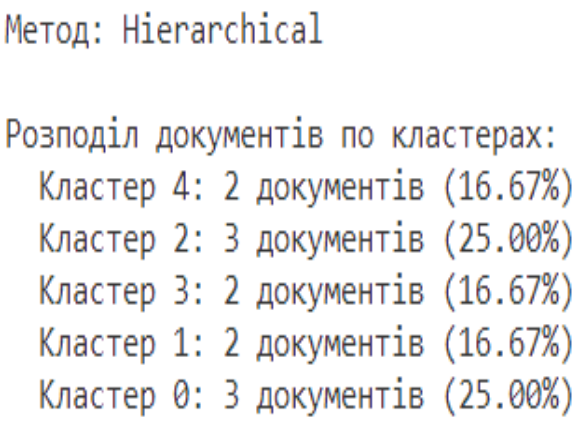

Рисунок 3.8 – Розподіл документів по кластерах для метода Hierarchical

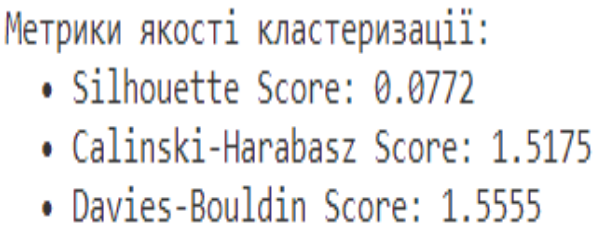

Рисунок 3.9 – Метрики якості кластеризації для метода Hierarchical

```
Приклади документів у кластерах:
  Кластер 4:
    - Машинне навчання є важливою галуззю штучного інтелекту...
    - Технології штучного інтелекту розвиваються дуже швидко...
  Кластер 2:
    - Глибоке навчання використовує нейронні мережі для аналізу даних...
    - Комп'ютерне бачення використовує глибоке навчання...
  Кластер 3:
    - Штучний інтелект змінює сучасний світ технологій...
    - Штучний інтелект підтримує автоматизацію процесів...
  Кластер 1:
    - Нейронні мережі дозволяють комп'ютерам навчатися як люди...
    - Нейронні мережі застосовуються в різних галузях...
  Кластер 0:
    - Алгоритми машинного навчання покращують точність прогнозування...
    - Машинне навчання має багато практичних застосувань...
```

Рисунок 3.10 – Приклади документів у кластерах для метода Hierarchical

Після проведених експериментів для обох згаданих методів доцільно провести та надати порівняльну характеристику. Отже, порівняння метрик класифікації наведено на рисунку 3.11.

Розподіл документів по кластерах наведено на рисунку 3.12.

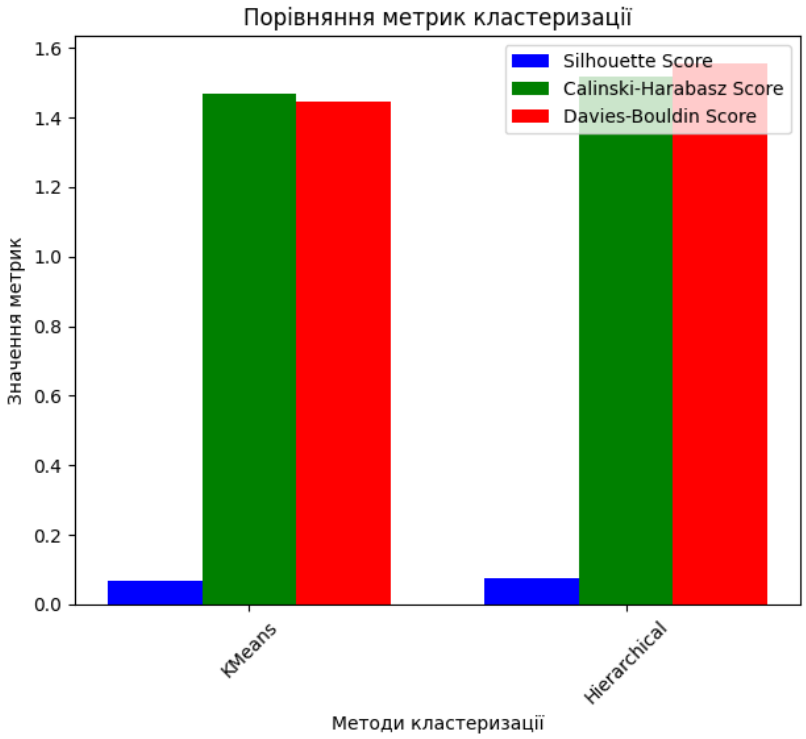

Рисунок 3.11 – Порівняння метрик кластеризації

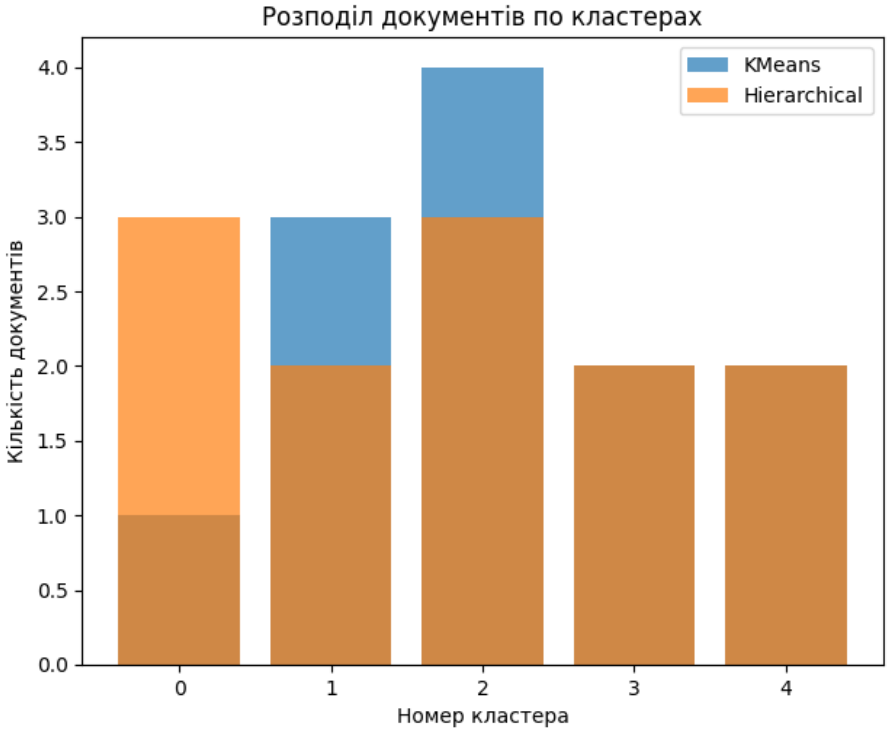

Рисунок 3.12 – Розподіл документів по кластерах

На рисунку 3.13 наведено 2D-проєкцію кластерів.

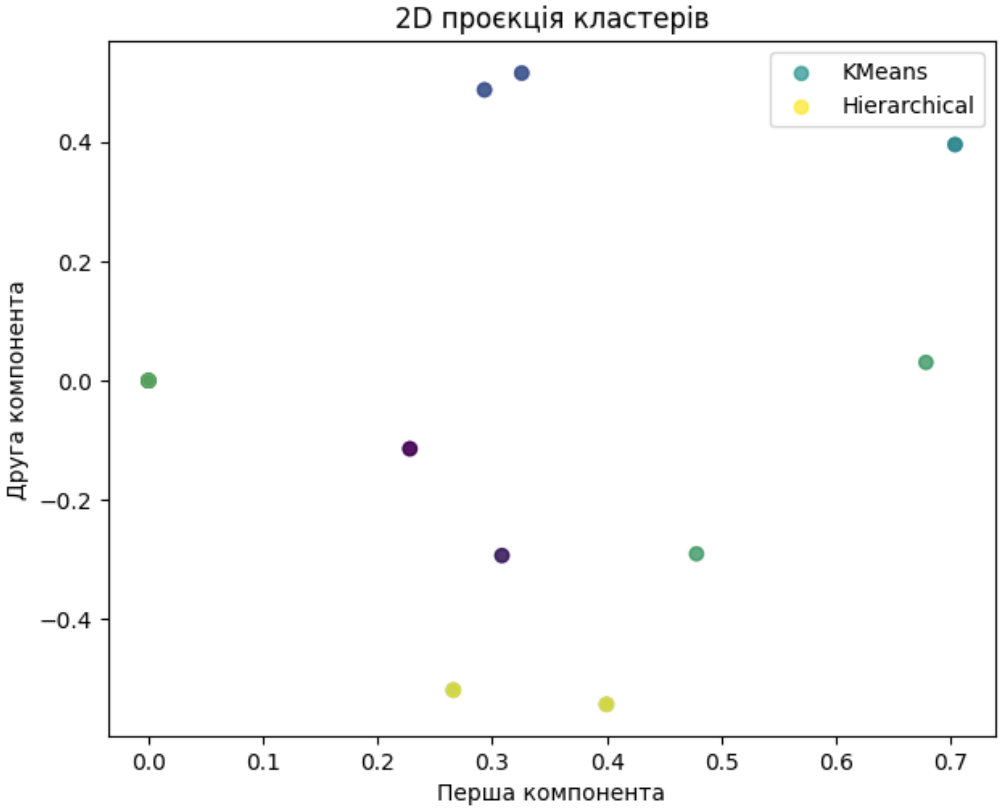

Рисунок 3.13 – 2D-проєкція кластерів

Останніми візуальними характеристиками доцільно представити відстані між кластерами у вигляді теплових карт (матриць). На рисунку 3.14 наведено таку матрицю для метода KMeans, а на рисунку 3.15 наведено матрицю для метода Hierarchical.

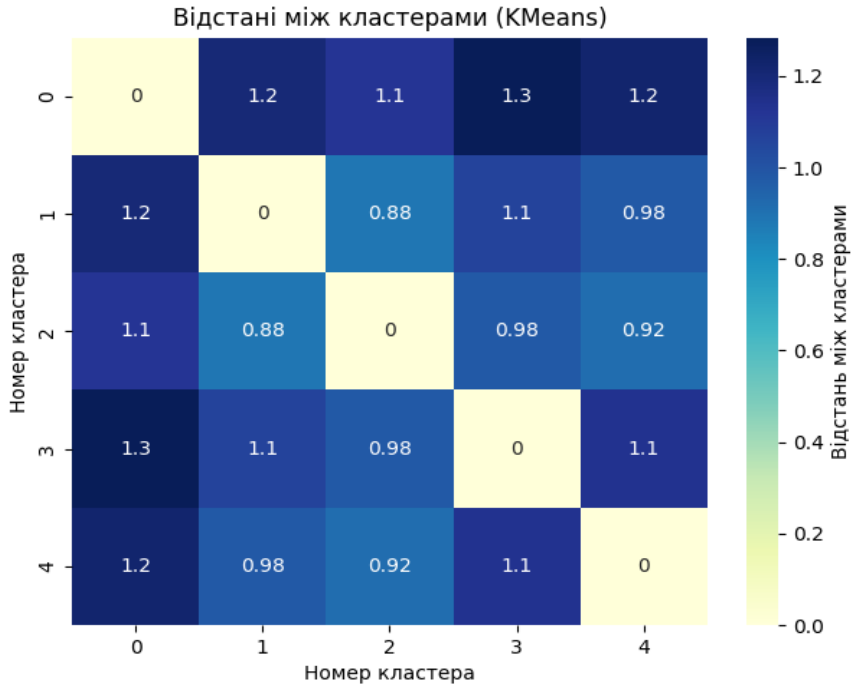

Рисунок 3.14 – Матриця відстаней між кластерами для метода KMeans

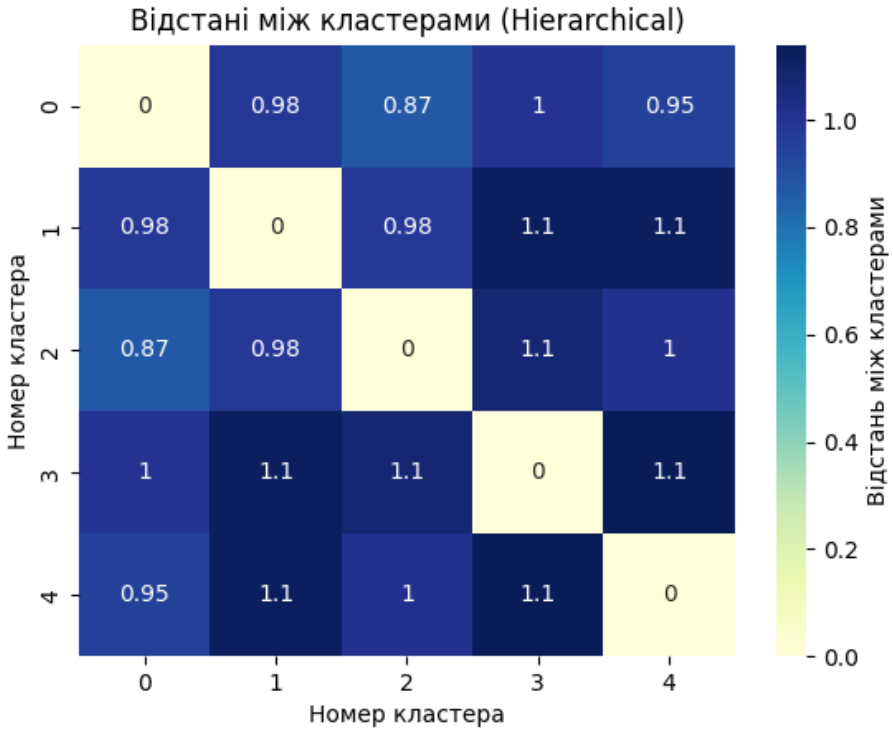

Рисунок 3.15 – Матриця відстаней між кластерами для метода Hierarchical

3.6 Висновки до третього розділу

В розділі прикладної частини було продемонстровано ефективність різних підходів, таких як метод K-середніх та ієрархічна кластеризація, що дозволило структурувати великі корпуси текстів. Результати показали, що правильно обрані алгоритми та параметри кластеризації, включаючи попередню обробку текстових даних, векторизацію, а також застосування метрик якості, сприяють формуванню значущих груп текстів. Це підкреслює важливість комбінованого використання сучасних бібліотек, таких як NLTK та Scikit-learn, у поєднанні з інтуїтивною візуалізацією результатів. Зокрема, графіки розподілу кластерів та теплові карти міжкластерних відстаней забезпечують краще розуміння отриманих даних.

Таким чином, розроблена програма є надійним інструментом для аналізу текстових документів, що демонструє свою гнучкість, масштабованість та можливість адаптації до різноманітних завдань у сфері обробки природної мови та аналізу даних.

ВИСНОВКИ

У ході дослідження були вивчені основні принципи кластеризації як одного з ключових інструментів аналізу даних, а також розглянуті її можливості в контексті обробки текстової інформації. Результати роботи підтверджують актуальність застосування алгоритмів кластеризації для вирішення складних завдань, пов’язаних із великими обсягами текстових даних, зокрема для сегментації, групування документів за змістом, аналізу настроїв користувачів та побудови рекомендаційних систем.

У дослідженні було виконано комплексний аналіз сучасних методів кластеризації, таких як K-середні, ієрархічна кластеризація, DBSCAN та алгоритми, які базуються на глибокому навчанні. Ці методи розглядались у контексті їхньої ефективності, гнучкості, вимог до попередньої обробки даних та придатності до розв’язання специфічних задач кластеризації тексту. Зокрема, було доведено, що для досягнення високої якості кластеризації критично важливими є вибір методу векторизації тексту, наприклад, використання TF-IDF, Word2Vec чи BERT, а також застосування відповідних метрик подібності, таких як косинусна подібність або евклідова відстань.

Прикладна частина роботи була зосереджена на розробці програмного забезпечення для реалізації та тестування кластеризаційних алгоритмів на основі реальних корпусів текстових даних. Використання таких бібліотек, як Scikit-learn та NLTK, дозволило ефективно застосовувати алгоритми кластеризації до наборів тексту після їхньої попередньої обробки. Під час експериментів було підтверджено, що попередня підготовка текстових даних, яка включає видалення стоп-слів, нормалізацію та лематизацію, є критично важливою для забезпечення точності кластеризації. Було виявлено, що правильний вибір параметрів алгоритму, таких як кількість кластерів або значення щільності для DBSCAN, значною мірою впливає на якість отриманих результатів.

Окрему увагу приділено оцінці якості кластеризації. Використання показників, таких як силуетні бали, дозволило кількісно оцінити продуктивність алгоритмів і надати об’єктивне уявлення про зручність використання кожного з них для певних типів задач. Також було продемонстровано, що використання візуалізацій, таких як графіки розподілу кластерів, дає можливість не лише оцінити якість кластеризації, але й виявити додаткові моменти щодо структури даних.

Результати роботи показали, що найбільш ефективними є підходи, які інтегрують сучасні техніки обробки тексту з потужними алгоритмами кластеризації, особливо в поєднанні з трансформаторними моделями. Це дозволяє досягати високого рівня точності, навіть для даних із високим ступенем неоднорідності. Крім того, продемонстровано, що такі методи забезпечують гнучкість і адаптивність до різних прикладних задач, наприклад, у маркетингових дослідженнях, сегментації клієнтів чи створенні тематичних моделей.

Загалом, результати кваліфікаційної роботи мають значну теоретичну та практичну цінність. Вони дозволяють краще зрозуміти, як ефективно застосовувати сучасні алгоритми кластеризації для аналізу текстових даних, а також надають практичні рекомендації щодо оптимального вибору підходів для конкретних завдань. Отримані знання та програмні розробки можуть бути використані для створення автоматизованих систем аналізу тексту, що мають важливе значення для бізнесу, науки та інших галузей, де робота з великими обсягами текстових даних є ключовою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. S. Dhanalakshmi, S. Sathiyabama, and D. Ayyamuthukumar, “A Novel Text Clustering Approach Based on Bacterial Colony Optimization”, in *2023 7th Int. Conf. Electron., Communication Aerosp. Technol. (ICECA)*, Coimbatore, India, Nov. 22–24, 2023. IEEE, 2023. https://doi.org/10.1109/iceca58529.2023.10395857

2. X. Wang, L. Zhang, X. Zhang, and K. Xie, “Application of Improved DBSCAN Clustering Algorithm on Industrial Fault Text Data”, in *2020 IEEE 18th Int. Conf. Ind. Inform. (INDIN)*, Warwick, United Kingdom, Jul. 20–23, 2020. IEEE, 2020. https://doi.org/10.1109/indin45582.2020.9442093

3. L. Akritidis, M. Alamaniotis, A. Fevgas, and P. Bozanis, “Confronting Sparseness and High Dimensionality in Short Text Clustering via Feature Vector Projections”, in *2020 IEEE 32nd Int. Conf. Tools with Artif. Intell. (ICTAI)*, Baltimore, MD, USA, Nov. 9–11, 2020. IEEE, 2020. https://doi.org/10.1109/ictai50040.2020.00129

4. W. Hidayat and A. Yaqin, “Business Trends Based on News Portal Websites for Analysis of Big Data Using K-Means Clustering”, in *2019 Int. Conf. Inf. Commun. Technol. (ICOIACT)*, Yogyakarta, Indonesia, Jul. 24–25, 2019. IEEE, 2019. https://doi.org/10.1109/icoiact46704.2019.8938413

5. H. M. Fadhil, Z. Abdulnasser, and S. Mohammed, “Improve Data Mining Techniques with a High-Performance Cluster”, in *2022 Int. Conf. Comput. Appl. (ICCA)*, Cairo, Egypt, Dec. 20–22, 2022. IEEE, 2022. https://doi.org/10.1109/icca56443.2022.10039629

6. Y. Zhang, Y. Zhang, Q. Zhao, and W. Rao, “Automatic User Categorization Through Large Transaction Data”, in *2019 IEEE Int. Conf. Multimedia Expo (ICME)*, Shanghai, China, Jul. 8–12, 2019. IEEE, 2019. https://doi.org/10.1109/icme.2019.00056

7. M. Ben-Fares, N. Grozavu, P. Rastin, and P. Holat, “High Dimensional Data Stream Clustering using Topological Representation Learning”, in *2022 IEEE*

Symp. Ser. Comput. Intell. (SSCI), Singapore, Singapore, Dec. 4–7, 2022. IEEE, 2022. https://doi.org/10.1109/ssci51031.2022.10022090

8. Y. Zhou, H. Chang, X. Deng, and X. Lu, "Research on automatic text clustering method based on Improved PSO", in *2022 12th Int. Conf. Inf. Technol. Medicine Educ. (ITME)*, Xiamen, China, Nov. 18–20, 2022. IEEE, 2022. https://doi.org/10.1109/itme56794.2022.00012

9. K. Sreekala, M. Sivajyothi, D. Chiranjevi, G. Sunitha, and M. Swetha, "A Novel Integration of Hierarchical Clustering and Ensemble Classification Algorithms for News Classification", in *2024 Int. Conf. Cogn. Robot. Intell. Syst. (ICC – ROBINS)*, Coimbatore, India, Apr. 17–19, 2024. IEEE, 2024. https://doi.org/10.1109/icc-robins60238.2024.10533963

10. H. Xu, "Research on clustering algorithms in data mining", in *2022 3rd Int. Conf. Big Data, Artif. Intell. Internet Things Eng. (ICBAIE)*, Xi'an, China, Jul. 15–17, 2022. IEEE, 2022. https://doi.org/10.1109/icbaie56435.2022.9985831

11. W. Wang, X. Zhou, F. Chen, and B. Cao, "Sequential Minimax Search for Multi-Layer Gene Grouping", *IEEE Access*, vol. 7, pp. 102931–102940, 2019. https://doi.org/10.1109/access.2019.2924491

12. M. F. Hasani, Y. Heryadi, Y. Arifin, Lukas, and W. Suparta, "Density Based Spatial Clustering of Applications with Noise and Sentence Bert Embedding for Indonesian Utterance Clustering", in *2023 Int. Conf. Comput. Sci., Inf. Technol. Eng. (ICCoSITE)*, Jakarta, Indonesia, Feb. 16, 2023. IEEE, 2023. https://doi.org/10.1109/iccosite57641.2023.10127683

13. M. Lu, J. Yin, K. Wang, and L. Nie, "A Multi-View Clustering Algorithm for Short Text", in *2024 IEEE 40th Int. Conf. Data Eng. (ICDE)*, Utrecht, Netherlands, May 13–16, 2024. IEEE, 2024. https://doi.org/10.1109/icde60146.2024.00107

14. H. Singh, S. Kaur, and C. Kaushal, "Text Document clustering using partial Fractionation and Bisecting K-means", in *2022 10th Int. Conf. Rel., Infocom*

Technol. Optim. (Trends Future Directions) (ICRITO), Noida, India, Oct. 13–14, 2022. IEEE, 2022. https://doi.org/10.1109/icrito56286.2022.9964710

15. Y. Chen, Q. Wang, X. Cai, and N. Wang, "A New Text Mining Method of Dispatching Operation Ticket System Based on Graph Partition Spectral Clustering Algorithm", in *2023 6th Int. Conf. Energy, Elect. Power Eng. (CEEPE)*, Guangzhou, China, May 12–14, 2023. IEEE, 2023. https://doi.org/10.1109/ceepe58418.2023.10166981

16. S. Yang, G. Huang, and B. Cai, "Discovering Topic Representative Terms for Short Text Clustering", *IEEE Access*, vol. 7, pp. 92037–92047, 2019. https://doi.org/10.1109/access.2019.2927345

ДОДАТКИ

Додаток А

Лістинг програмного коду

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.decomposition import PCA, TruncatedSVD
from sklearn.metrics import (
    silhouette_score,
    calinski_harabasz_score,
    davies_bouldin_score
)
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
from collections import Counter
import warnings
warnings.filterwarnings('ignore')

# Завантаження додаткових ресурсів NLTK
nltk.download('punkt', quiet=True)
nltk.download('stopwords', quiet=True)
```

```
class AdvancedTextClusterAnalyzer:
    def __init__(self, documents):
        """
        Розширена ініціалізація аналізатора кластеризації текстових документів.
        """
        # Фільтрація порожніх документів
        self.documents = [doc for doc in documents if doc and isinstance(doc, str) and len(doc.strip()) > 0]

        if len(self.documents) == 0:
            raise ValueError("Немає коректних текстових документів для аналізу!")

        self.preprocessed_docs = []
        self.vectorizer = None
        self.feature_matrix = None
        self.clustering_results = {}

    def preprocess_text(self, text):
        """
        Поглиблена попередня обробка тексту.
        """
        text = text.lower()
        text = re.sub(r'[^a-яа-z\s]', '', text)
        tokens = word_tokenize(text)
        stop_words = set(stopwords.words('russian') + stopwords.words('english'))
        tokens = [token for token in tokens if token not in stop_words]
        return ' '.join(tokens)
```

```
    def prepare_documents(self):
        """
        Підготовка та векторизація документів.
        """
        self.preprocessed_docs = [self.preprocess_text(doc) for doc in
self.documents]

        if len(self.preprocessed_docs) == 0:
            raise ValueError("Після попередньої обробки не залишилось
документів!")

        self.vectorizer = TfidfVectorizer(max_features=1000)
        self.feature_matrix = self.vectorizer.fit_transform(self.preprocessed_docs)

        if self.feature_matrix.shape[0] == 0:
            raise ValueError("Не вдалося створити матрицю ознак!")

    def perform_clustering(self, kmeans_clusters=min(5, max(2, 12//2))):
        """
        Виконання кластеризації різними методами з детальною аналітикою.
        """
        clustering_methods = {
            'KMeans': lambda: self._kmeans_clustering(n_clusters=kmeans_clusters),
            'Hierarchical': lambda:
self._hierarchical_clustering(n_clusters=kmeans_clusters)
        }

        try:
```

```
            for method_name, clustering_func in clustering_methods.items():
                result = clustering_func()
                result['method'] = method_name
                self.clustering_results[method_name] = result
        except Exception as e:
            print(f"Помилка при кластеризації: {e}")
            print(f"Кількість документів: {len(self.documents)}")
            print(f"Форма матриці ознак: {self.feature_matrix.shape}")

    def _kmeans_clustering(self, n_clusters=None):
        """
        Кластеризація методом K-means з розширеною аналітикою.
        """
        if n_clusters is None:
            n_clusters = min(5, max(2, len(self.documents)//2))

        kmeans = KMeans(n_clusters=n_clusters, random_state=42)
        labels = kmeans.fit_predict(self.feature_matrix)

        return {
            'labels': labels,
            'centroids': kmeans.cluster_centers_,
            'silhouette_score': silhouette_score(self.feature_matrix, labels),
            'calinski_harabasz_score':
calinski_harabasz_score(self.feature_matrix.toarray(), labels),
            'davies_bouldin_score': davies_bouldin_score(self.feature_matrix.toarray(),
labels)
        }
```

```
    def _hierarchical_clustering(self, n_clusters=None):
        """
        Ієрархічна кластеризація з розширеною аналітикою.
        """
        if n_clusters is None:
            n_clusters = min(5, max(2, len(self.documents)//2))

        hierarchical = AgglomerativeClustering(n_clusters=n_clusters)
        labels = hierarchical.fit_predict(self.feature_matrix.toarray())

        return {
            'labels': labels,
            'silhouette_score': silhouette_score(self.feature_matrix, labels),
            'calinski_harabasz_score':
calinski_harabasz_score(self.feature_matrix.toarray(), labels),
            'davies_bouldin_score': davies_bouldin_score(self.feature_matrix.toarray(),
labels)
        }

    def visualize_advanced_clustering_analysis(self):
        """
        Розширена графічна аналітика кластеризації.
        """
        plt.figure(figsize=(20, 15))
        plt.subplots_adjust(hspace=0.4, wspace=0.3)

        # 1. Порівняння метрик кластеризації
        plt.subplot(2, 3, 1)
        metrics = {
```

```
        'Silhouette Score': [],
        'Calinski-Harabasz Score': [],
        'Davies-Bouldin Score': []
    }
    methods = []

    for method, result in self.clustering_results.items():
        methods.append(method)
        metrics['Silhouette Score'].append(
            result.get('silhouette_score', 0)
        )
        metrics['Calinski-Harabasz Score'].append(
            result.get('calinski_harabasz_score', 0)
        )
        metrics['Davies-Bouldin Score'].append(
            result.get('davies_bouldin_score', 0)
        )

    x = np.arange(len(methods))
    width = 0.25

    plt.bar(x - width, metrics['Silhouette Score'], width, label='Silhouette Score',
color='blue')
    plt.bar(x, metrics['Calinski-Harabasz Score'], width, label='Calinski-Harabasz
Score', color='green')
    plt.bar(x + width, metrics['Davies-Bouldin Score'], width, label='Davies-
Bouldin Score', color='red')

    plt.xlabel('Методи кластеризації')
```

```
plt.ylabel('Значення метрик')
plt.title('Порівняння метрик кластеризації')
plt.xticks(x, methods, rotation=45)
plt.legend(loc='best')

# 2. Розподіл документів по кластерах
plt.subplot(2, 3, 2)
for method, result in self.clustering_results.items():
    labels = result['labels']
    label_counts = Counter(labels)

    plt.bar(list(label_counts.keys()), list(label_counts.values()),
            label=method, alpha=0.7)

plt.xlabel('Номер кластера')
plt.ylabel('Кількість документів')
plt.title('Розподіл документів по кластерах')
plt.legend()

# 3. Зменшення вимірності та 2D проєкція
plt.subplot(2, 3, 3)
svd = TruncatedSVD(n_components=2)

for method, result in self.clustering_results.items():
    labels = result['labels']
    reduced_features = svd.fit_transform(self.feature_matrix.toarray())

    plt.scatter(reduced_features[:, 0], reduced_features[:, 1],
                c=labels, label=method, alpha=0.7)
```

```
plt.xlabel('Перша компонента')
plt.ylabel('Друга компонента')
plt.title('2D проєкція кластерів')
plt.legend()

# 4. Теплова карта відстаней між кластерами
plt.subplot(2, 3, 4)
for method, result in self.clustering_results.items():
    labels = result['labels']
    distance_matrix = np.zeros((len(set(labels)), len(set(labels))))

    for i in range(len(set(labels))):
        for j in range(len(set(labels))):
            cluster_i_docs = self.feature_matrix[labels == i]
            cluster_j_docs = self.feature_matrix[labels == j]
            distance_matrix[i, j] = np.mean(np.linalg.norm(
                cluster_i_docs.mean(axis=0) - cluster_j_docs.mean(axis=0)
            ))

    sns.heatmap(distance_matrix, annot=True, cmap='YlGnBu',
                xticklabels=range(len(set(labels))),
                yticklabels=range(len(set(labels))),
                cbar_kws={'label': 'Відстань між кластерами'})
    plt.title(f'Відстані між кластерами ({method})')
    plt.xlabel('Номер кластера')
    plt.ylabel('Номер кластера')
    plt.show()
```

```
        # 5. Розподіл слів у кластерах
        plt.subplot(2, 3, 5)
        for method, result in self.clustering_results.items():
            labels = result['labels']
            cluster_terms = {}

            for i in range(len(set(labels))):
                cluster_docs = [self.preprocessed_docs[j] for j in range(len(labels)) if
labels[j] == i]
                all_words = ' '.join(cluster_docs).split()
                word_freq = Counter(all_words)
                cluster_terms[i] = word_freq.most_common(5)

            plt.bar(range(len(cluster_terms)),
                    [len(terms) for terms in cluster_terms.values()],
                    label=method, alpha=0.7)

        plt.xlabel('Кластер')
        plt.ylabel('Кількість унікальних термінів')
        plt.title('Різноманітність термінів у кластерах')
        plt.legend()

        plt.tight_layout()
        plt.show()

    def generate_cluster_report(self):
        """
        Генерація детального звіту про кластеризацію.
        """
```

```
        print("Звіт про кластеризацію текстових документів:\n")

        for method, result in self.clustering_results.items():
            print(f"Метод: {method}")

            # Аналіз розподілу документів по кластерах
            labels = result['labels']
            label_counts = Counter(labels)

            print("\nРозподіл документів по кластерах:")
            for cluster, count in label_counts.items():
                percentage = (count / len(labels)) * 100
                print(f"  Кластер {cluster}: {count} документів ({percentage:.2f}%)")

            # Виведення метрик
            print("\nМетрики якості кластеризації:")
            if 'silhouette_score' in result:
                print(f"  • Silhouette Score: {result['silhouette_score']:.4f}")
            if 'calinski_harabasz_score' in result:
                print(f"  • Calinski-Harabasz Score:
{result['calinski_harabasz_score']:.4f}")
            if 'davies_bouldin_score' in result:
                print(f"  • Davies-Bouldin Score: {result['davies_bouldin_score']:.4f}")

            # Приклади документів по кластерах
            cluster_docs = {}
            for i, label in enumerate(labels):
                if label not in cluster_docs:
                    cluster_docs[label] = []
```

```
            cluster_docs[label].append(self.documents[i])

        print("\nПриклади документів у кластерах:")
        for cluster, docs in cluster_docs.items():
            print(f"  Кластер {cluster}:")
            for doc in docs[:2]:
                print(f"    - {doc[:100]}...")
        print("\n" + "="*50 + "\n")

def main():
    # Приклад текстових документів
    documents = [
        "Машинне навчання є важливою галуззю штучного інтелекту",
        "Глибоке навчання використовує нейронні мережі для аналізу даних",
        "Штучний інтелект змінює сучасний світ технологій",
        "Нейронні мережі дозволяють комп'ютерам навчатися як люди",
        "Алгоритми машинного навчання покращують точність прогнозування",
        "Комп'ютерне бачення використовує глибоке навчання",
        "Технології штучного інтелекту розвиваються дуже швидко",
        "Машинне навчання має багато практичних застосувань",
        "Data Science допомагає аналізувати великі обсяги інформації",
        "Нейронні мережі застосовуються в різних галузях",
        "Штучний інтелект підтримує автоматизацію процесів",
        "Комп'ютерне навчання революціонізує сучасні технології"
    ]

    try:
        # Створення розширеного аналізатора
        analyzer = AdvancedTextClusterAnalyzer(documents)
```

```
        # Підготовка та кластеризація
        analyzer.prepare_documents()
        analyzer.perform_clustering()

        # Генерація звіту
        analyzer.generate_cluster_report()

        # Графічна аналітика
        analyzer.visualize_advanced_clustering_analysis()

    except Exception as e:
        print(f"Сталася помилка: {e}")

if __name__ == "__main__":
    main()
```