МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ГРИЦАЙ Ростислав Ігорович

Модуль підбору тематики наукових статей на основі NLP / Module for selecting topics for scientific articles based on NLP

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Дипломний проект

Виконав студент групи КН-41
Р.І. Грицай

Науковий керівник:
к.т.н., доцент Т.В. Лендюк

Дипломний проект допущено до захисту
«___»_____ 2023 р.

Завідувач кафедри
_____ М.П. Комар

Тернопіль – 2023

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2022р.

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Грицай Ростиславу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема проекту: Модуль підбору тематики наукових статей на основі NLP / Module for selecting topics for scientific articles based on NLP
керівник проекту к.т.н., доцент Т.В. Лендюк

затверджені наказом по університету від 08 грудня 2022 р. № 491.

2. Строк подання студентом закінченого проекту 01 червня 2023 р.

3. Вихідні дані до проекту: технічне завдання.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- проаналізувати передумов для інтелектуального аналізу тексту та існуючі рішення.
- розробити структуру обробки інтелектуального аналізу тексту.
- розробити алгоритм обробки та розподілу набору даних для текстового аналізу.
- розробити алгоритм tf-idf для підбору тематики наукових статей та алгоритм тренування моделі doc2vec.
- розробити алгоритм кластеризації документів для підбору тематики наукових статей.
- провести попередню обробку даних для інтелектуального аналізу тексту.
- натренувати моделі doc2vec для підбору тематики наукових статей.
- провести кластеризацію документів для підбору тематики наукових статей.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

- структура обробки інтелектуального аналізу тексту.
- алгоритм обробки та розподілу набору даних для текстового аналізу.
- алгоритм tf-idf для підбору тематики наукових статей та алгоритм тренування моделі doc2vec.
- алгоритм кластеризації документів для підбору тематики наукових статей.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Н. контроль	к.т.н., доцент Т.В. Лендюк		

7. Дата видачі завдання 08 грудня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1	Аналіз предметної області і постановка задачі дослідження	30.12.2022	
2	Алгоритмічне забезпечення підбору тематики наукових статей	24.03.2023	
3	Програмно-технологічне забезпечення підбору тематики наукових статей	12.05.2023	
4	Повне завершення та оформлення дипломного проекту	01.06.2023	

Студент _____ Р.І. Грицай
(підпис)

Керівник проекту_____ Т.В. Лендюк
(підпис)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту: 75 с., 13 рис., 1 табл., 2 додатки, 31 джерел.

Метою дипломного проекту є розробка та реалізація модуля підбору тематики наукових статей на основі NLP.

Об'єктом дослідження є модуль підбору тематики наукових статей на основі NLP.

Предметом дослідження є процес підбору тематики наукових статей на основі NLP.

Розроблено та досліджено програмне забезпечення для пошуку та вибору наукових статей для користувачів.

Розроблений модуль підбору тематики наукових статей на основі NLP дозволить користувачам швидше та ефективніше знаходити статті, які найкраще відповідають їхнім потребам та інтересам. Це може бути корисним для студентів, науковців, викладачів, дослідників та інших осіб, які займаються науковою або академічною діяльністю.

NLP, СИСТЕМА ОБРОБКИ ІНФОРМАЦІЇ, НАУКОВІ СТАТТІ, ВЕЛИКІ ДАНІ.

ABSTRACT

The bachelor's thesis report: 75 pages, 13 figures, 1 table, 2 appendices, 31 references.

The aim of the diploma project is to develop and implement a module for selecting the topics of scientific articles based on NLP.

The object of research is the module for selecting the topics of scientific articles based on NLP.

The subject of research is the process of selecting the topics of scientific articles based on NLP.

Software has been developed and investigated for searching and selecting scientific articles for users.

The developed module for selecting the topics of scientific articles based on NLP will allow users to find articles that best meet their needs and interests faster and more efficiently. This can be beneficial for students, researchers, professors, scholars, and other individuals engaged in scientific or academic activities.

NLP, INFORMATION PROCESSING SYSTEM, SCIENTIFIC ARTICLES, BIG DATA.

ТЕХНІЧНЕ ЗАВДАННЯ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

1.1 Модуль підбору тематики наукових статей на основі NLP.

1.2 Область застосування – розроблений модуль може бути застосований у різних сферах, включаючи наукові дослідження, академічну діяльність, освіту, технологічний розвиток та інновації.

2. ОСНОВА ДЛЯ РОЗРОБЛЕННЯ

Основою для розроблення є завдання на дипломний проект, затверджене кафедрою інформаційно-обчислювальних систем і управління факультету комп'ютерних інформаційних технологій Західноукраїнського національного університету.

3. ПРИЗНАЧЕННЯ РОЗРОБЛЕНОГО КОМПЛЕКСУ

Метою дипломного проекту є розробка та реалізація модуля підбору тематики наукових статей на основі NLP.

4. ДЖЕРЕЛА РОЗРОБЛЕННЯ

Джерелами даної розробки є матеріали навчальної і реферативної літератури, технічна документація, науково-дослідні статті, журнали, Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Основні функціональні вимоги до програмної системи:

– вибір даних (пакетних даних / даних у реальному часі) з різних архітектур великих даних;

– провести попередню обробку даних для інтелектуального аналізу тексту.

– натренувати моделі doc2vec для підбору тематики наукових статей.

– провести кластеризацію документів для підбору тематики наукових статей.

5.2 Вимоги до апаратних засобів:

– кластер запускається на фізичній машині Macbook Pro з 16 гігабайт оперативної пам’яті та процесором 2.3 GHz Intel Core i5.

5.3 Вимоги до програмних засобів:

– для розробки програмне забезпечення - Python 3.7;

– для створення графічного інтерфейсу користувача використано – tkinter, Adobe XD;

– для реалізації моделей навчання – фреймворк TF-IDF, Doc2Vec.

6. ПОРЯДОК КОНТРОЛЮ

6.1 Представлення дипломного проекту на попередній захист.

6.2 Представлення дипломного проекту на захист.

Завдання прийняв до виконання _____ Р.І. Грицай

(підпис) (прізвище та ініціали)

Керівник дипломного проекту_____ Т.В. Лендюк

(підпис) (прізвище та ініціали)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

1. HTML: HyperText Markup Language - мова розмітки гіпертексту, використовується для створення веб-сторінок.
2. NLP: Natural Language Processing - обробка природної мови, галузь штучного інтелекту, яка займається обробкою та розумінням мови, яку використовують люди.
3. TF-IDF: Term Frequency-Inverse Document Frequency - статистичний метод для оцінки важливості слова в контексті документа або колекції документів.
4. PCA: Principal Component Analysis - метод аналізу даних, який використовується для зменшення розмірності даних шляхом проектування їх на простір меншої кількості вимірів, зберігаючи при цьому максимальну дисперсію.
5. Doc2Vec: Document-to-Vector - алгоритм для отримання числових представлень (векторів) документів, який базується на моделі Word2Vec і дозволяє прив'язати семантичні значення до текстових документів.
6. CLEF 2003: Cross-Language Evaluation Forum 2003 - міжнародне форумне змагання, спрямоване на покращення систем з пошуку інформації та обробки природної мови в багатомовному контексті.
7. LA-PDF: Language Analysis for PDF - алгоритм аналізу мови, спеціально призначений для обробки і аналізу текстових документів у форматі PDF.

ВСТУП

Актуальність теми. Актуальність теми полягає в тому, що в сучасному світі, де інформація надходить величезним потоком, знайти потрібну інформацію може бути складною задачею. Багато людей шукають наукові статті для роботи, навчання, наукових досліджень або просто для задоволення своєї цікавості. Однак, при пошуку важко знайти статті, які б максимально відповідали б конкретним потребам користувача.

Тому модуль підбору тематики наукових статей на основі NLP може значно спростити і поліпшити пошук статей. За допомогою природних мовних технологій (NLP), модуль може аналізувати запити користувача та використовувати цю інформацію для підбору статей, що відповідають їхнім потребам.

Крім того, застосування NLP може допомогти у поліпшенні пошуку статей за ключовими словами, темами та стилістичними особливостями. Наприклад, модуль може використовувати аналіз тексту, щоб визначити тематику статті та порівняти її з запитом користувача.

Отже, модуль підбору тематики наукових статей на основі NLP може бути корисним для різних категорій користувачів, які шукають наукові статті, забезпечуючи їм більш точний та зручний пошук.

Метою дипломного проекту є розробка та реалізація модуля підбору тематики наукових статей на основі NLP.

Для досягнення поставленої мети було необхідно провести наступні теоретичні розробки та виконати послідовні завдання:

1. проаналізувати передумов для інтелектуального аналізу тексту та існуючі рішення.
2. розробити структуру обробки інтелектуального аналізу тексту.
3. розробити алгоритм обробки та розподілу набору даних для текстового аналізу.

4. розробити алгоритм tf-idf для підбору тематики наукових статей та алгоритм тренування моделі doc2vec.
5. розробити алгоритм кластеризації документів для підбору тематики наукових статей.
6. провести попередню обробку даних для інтелектуального аналізу тексту.
7. натренувати моделі doc2vec для підбору тематики наукових статей.
8. провести кластеризацію документів для підбору тематики наукових статей

Об'єктом дослідження є модуль підбору тематики наукових статей на основі NLP.

Предметом дослідження є процес підбору тематики наукових статей на основі NLP.

Методи дослідження включатимуть аналіз літератури з області NLP та підбору статей, використання алгоритмів обробки природньої мови для аналізу текстів статей та виявлення їхньої тематики, розробку та впровадження алгоритмів порівняння текстів, створення та апробацію прототипу модуля підбору статей на основі NLP, збір та аналіз даних для оцінки ефективності модуля, проведення користувацьких тестувань та оцінку результатів**.**

Практичне значення полягає в поліпшенні процесу пошуку та вибору наукових статей для користувачів. Розроблений модуль підбору тематики наукових статей на основі NLP дозволить користувачам швидше та ефективніше знаходити статті, які найкраще відповідають їхнім потребам та інтересам. Це може бути корисним для студентів, науковців, викладачів, дослідників та інших осіб, які займаються науковою або академічною діяльністю.

Структура та обсяг роботи. Дипломна робота складається із вступу, трьох розділів, висновків, списку використаних джерел. Повний обсяг роботи

становить 75 ст. комп’ютерного тексту, який включає 13 рисунки та 1 таблиця. Список використаних джерел із 31 найменувань викладено на 5 сторінках.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Передумови аналізу тексту

Сьогодні майже вся наявна інформація в різних установах (наприклад, уряді, бізнесі, промисловості та інших) зберігається в електронних документах, у яких вона містить напівструктуровані дані. У цих документах «реферат» є прикладом неструктурованого текстового компонента. Тоді як прикладами структурованих полів у документі є: ім’я автора, дата публікації, назва та категорія [1]. Дослідження, проведене [2], показало, що інтелектуальний аналіз тексту став однією з модних галузей, яка була включена в кілька галузей досліджень, таких як комп’ютерна лінгвістика, пошук інформації (IP) та інтелектуальний аналіз даних. Інтелектуальний аналіз тексту відрізняється від аналізу даних [3]. Інтелектуальний аналіз даних зосереджений на виявленні цікавих шаблонів у великих базах даних, а не на текстовій інформації [4]. Методології відновлення інформації, такі як техніка індексування тексту, були розроблені для обробки неструктурованих документів. У звичайних дослідженнях передбачається, що користувач здебільшого шукає відомі терміни, які раніше використовувалися або були написані кимось іншим. Основна проблема полягає в тому, що результати пошуку не відповідають вимогам користувача. Одним із рішень є використання текстового аналізу, щоб знайти релевантну інформацію, яка поки що не вказана явно й не записана. Процедура текстового майнінгу починається зі збору документів через різні ресурси. Конкретний документ буде відновлено за допомогою інструменту аналізу тексту та шляхом перевірки його формату та наборів символів; він буде попередньо оброблений цим інструментом. Потім документ проходив би етап аналізу тексту. Аналіз тексту включає семантичний аналіз, призначений для отримання якісної

інформації через текст. Доступні різні методи аналізу тексту. Залежно від мети організації можна використовувати різні методи. У деяких випадках методи аналізу тексту повторюються, поки інформація не буде вилучена. Результати можуть зберігатися в інформаційній системі управління, яка надає великий обсяг важливої інформації для користувача цієї системи.

Інтелектуальний аналіз тексту спрямований на виявлення інформації, яка раніше не була розпізнана, шляхом її автоматичного вилучення з різних текстових джерел. Структуровані дані можна обробляти за допомогою інструментів інтелектуального аналізу даних, тоді як неструктуровані або напівструктуровані набори даних, як-от повнотекстові документи, електронні листи та файли HTML, можна обробляти за допомогою інтелектуального аналізу даних. Як правило, інформація зберігається в природній формі, відомій як текст. Текстовий майнінг не схожий на веб-майнінг. Коли користувач щось досліджує в Інтернеті, це означає, що це відомо раніше і це було написано кимось іншим [5]. Наприклад, в електронній комерції основною проблемою веб-майнінгу є купівля всіх матеріалів, які не відповідають пошуковому запиту користувача, і вони не відображатимуть невідому (приховану або неявну) інформацію, тоді як основною метою текстового майнінгу є пошук вивести невідому інформацію [6]; те, що ніким не визнано.

Дані — це основний тип інформації, який потрібно впорядковувати та видобувати для генерації знань. Виявлення закономірностей і тенденцій на основі величезних даних є серйозним завданням. Правильне виявлення невідомих тенденцій і закономірностей у базах даних є основною метою інтелектуального аналізу даних. Це метод, у якому необхідна попередня обробка даних перед застосуванням будь-якого іншого методу. Багато підходів, таких як кластеризація, класифікація та дерева рішень, задіяні в інтелектуальному аналізі даних. Вся текстова інформація зберігається в електронному вигляді на персональних комп’ютерах клієнта або на веб-

сервері. Завдяки постійному зростанню апаратних пристроїв зберігання даних будь-який комп'ютер або ноутбук має можливість зберігати величезну кількість даних. Створення нової інформації може бути простим, у той час як знайти відповідну інформацію з величезної кількості даних – складно. Щоб отримати відповідну інформацію, знання або шаблони з різних джерел, які знаходяться в неструктурованій формі, можна використовувати техніку аналізу тексту. Загальна структура аналізу тексту включає два послідовних етапи: уточнення тексту та дистиляція знань. Під час уточнення тексту текстові документи вільної форми перетворюються на проміжну форму, тоді як при дистиляції знань шаблони або знання виводяться з проміжної форми. Проміжна форма (IF) може бути або напівструктурованою, як ілюстрація теоретичного графа, або структурованою, як ілюстрація реляційних даних. IF може бути заснованим на документі, де кожна сутність символізує документ, він може бути заснованим на концепції, де кожна одиниця символізує об'єкт або концепцію інтересу в конкретній області. або шаблони з різних джерел, які знаходяться в неструктурованій формі, можна використовувати техніку аналізу тексту. Загальна структура видобутку тексту включає два послідовних етапи: уточнення тексту та дистиляція знань. Під час уточнення тексту текстові документи вільної форми перетворюються на проміжну форму, тоді як при дистиляції знань шаблони або знання виводяться з проміжної форми. Проміжна форма (IF) може бути або напівструктурованою, як ілюстрація теоретичного графа, або структурованою, як ілюстрація реляційних даних. IF може бути заснованим на документі, де кожна сутність символізує документ, він може бути заснованим на концепції, де кожна одиниця символізує об'єкт або концепцію інтересу в конкретній області. або шаблони з різних джерел, які знаходяться в неструктурованій формі, можна використовувати техніку аналізу тексту. Загальна структура видобутку тексту включає два послідовних етапи: уточнення тексту та дистиляція знань. Під час уточнення тексту текстові документи вільної форми

перетворюються на проміжну форму, тоді як при дистиляції знань шаблони або знання виводяться з проміжної форми. Проміжна форма (IF) може бути або напівструктурованою, як ілюстрація теоретичного графа, або структурованою, як ілюстрація реляційних даних. IF може бути заснованим на документі, де кожна сутність символізує документ, він може бути заснованим на концепції, де кожна одиниця символізує об’єкт або концепцію інтересу в конкретній області. Загальна структура аналізу тексту включає два послідовних етапи: уточнення тексту та дистиляція знань. Під час уточнення тексту текстові документи вільної форми перетворюються на проміжну форму, тоді як при дистиляції знань шаблони або знання виводяться з проміжної форми. Проміжна форма (IF) може бути або напівструктурованою, як ілюстрація теоретичного графа, або структурованою, як ілюстрація реляційних даних. IF може бути заснованим на документі, де кожна сутність символізує документ, він може бути заснованим на концепції, де кожна одиниця символізує об’єкт або концепцію інтересу в конкретній області. Загальна структура аналізу тексту включає два послідовних етапи: уточнення тексту та дистиляція знань. Під час уточнення тексту текстові документи вільної форми перетворюються на проміжну форму, тоді як при дистиляції знань шаблони або знання виводяться з проміжної форми. Проміжна форма (IF) може бути або напівструктурованою, як ілюстрація теоретичного графа, або структурованою, як ілюстрація реляційних даних. IF може бути заснованим на документі, де кожна сутність символізує документ, він може бути заснованим на концепції, де кожна одиниця символізує об’єкт або концепцію інтересу в конкретній області. Проміжна форма (IF) може бути або напівструктурованою, як ілюстрація теоретичного графа, або структурованою, як ілюстрація реляційних даних. IF може бути заснованим на документі, де кожна сутність символізує документ, він може бути заснованим на концепції, де кожна одиниця символізує об’єкт або концепцію інтересу в конкретній

області. Проміжна форма (IF) може бути або напівструктурованою, як ілюстрація теоретичного графа, або структурованою, як ілюстрація реляційних даних. IF може бути заснованим на документі, де кожна сутність символізує документ, він може бути заснованим на концепції, де кожна одиниця символізує об’єкт або концепцію інтересу в конкретній області.

Різні дослідницькі області, методи та моделі залучаються до різних дослідницьких областей. Найгарячіші теми дослідницьких областей є основною темою багатьох наукових робіт. Результати дослідження певної області можуть впливати на інші галузі досліджень, оскільки деякі галузі досліджень можуть мати схожі теми. Ці теми досліджень завжди обговорюють такий перспективний напрямок досліджень, який варто вивчати. Таким чином, у цьому дослідженні визначено тенденцію міждоменного розвитку. У цьому дослідженні за допомогою методів інтелектуального аналізу тексту досліджувалися лонгітюдні тенденції академічних статей у Mobile Leaning. Ми відновили та дослідили (300) реферованих журнальних статей і матеріалів конференцій з різних автентичних баз даних.

Основними цілями цього дослідження є: (1) Використання методів аналізу тексту для визначення тем наукового тексту, пов’язаного з дослідженнями машинного навчання, та встановлення ієрархічного та еволюційного зв’язку між цими темами. (2) Використання інструментів візуалізації для представлення як тем, так і зв’язків між ними як зручного способу допомогти користувачам визначити відповідні теми.

1.2 Аналіз існуючих рішень у сфері аналізу тексту

Багато дослідницьких робіт зробили внесок у сферу IO завдяки використанню різних методів. Основною метою цих досліджень було визначення того, як можна використовувати різні процедури інтелектуального аналізу тексту, оскільки набори структурованих даних існують у форматі

текстового документа. Ця частина починається з визначення теми дослідження, оцінки попередніх досліджень, а потім застосовуються основні методи за допомогою вилучення інформації та аналізу тексту. Щоб визначити тему кожного наукового напряму та розробити еволюційний та ієрархічний зв'язок між цими темами, [35] використано метод текстового майнінгу. Теми представлені за допомогою засобів візуалізації. Крім того, ці інструменти використовуються для того, щоб показати зв'язок між цими темами та запропонувати інтерактивні функції, щоб користувачі могли ефективно знаходити міждоменні теми та знати тенденції міждоменного дослідження.

Молошников та ін. [36] розроблено алгоритм пошуку документів з певної теми в залежності від обраної довідкової колекції документів. Крім того, також розроблено контекстно-семантичний граф для тем візуалізації в результатах пошуку. Алгоритм залежить від об'єднання групи ентропійних, імовірнісних і семантичних розробників для видобутку зважених ключових слів і набору слів, які пояснюють вказану тему. Результати показали, що середня точність становить 99%, а відкликання – 84%. Також була створена унікальна техніка для побудови графіків на основі алгоритму, вміє видаляти ключові фрази з вагами. Він пропонує можливість показати розташування підтем у величезних наборах документів у формі компактного графіка.

Щоб запропонувати посилання на додаткові дослідження інших дослідників, [37] обговорив статус досліджень технології інтелектуального аналізу тексту під час її використання в біомедичній галузі, що охоплює 10 років. Література з аналізу біомедичних текстів, включена в SCI з 2004 по 2013 роки, була відновлена, відфільтрована, а потім досліджена з точки зору дослідницьких установ, щорічних змін, областей досліджень, місцевого розповсюдження, джерел журналів і ключових слів. Спостерігається значне збільшення кількості світової біомедичної літератури з аналізу текстів. Серед цієї глобальної літератури величезний відсоток займає література, пов'язана з розпізнаванням іменованих сутностей, виділенням зв'язків сутностей,

категоризацією тексту, кластеризацією тексту, виділенням абревіатур і аналізом спільного входження. Дослідження, проведені в США та Великобританії, вважаються присутніми на першому місці.

Для виділення міжмовних кластерів через багатомовні документи залежно від замкнутого аналізу концепцій і векторної моделі [38] був запропонований новий статистичний підхід. Методи аналізу формальних понять використовуються для вилучення закритих понять із подібних корпусів, а пізніше ці закриті поняття та векторні моделі використовуються для кластеризації та впорядкування багатомовних документів. Експериментальна оцінка проведена на наборі франко-англійських двомовних документів CLEF 2003. Результати показали, що взаємодія між векторною моделлю та формальним концептуальним аналізом є дуже корисною завдяки значній оцінці порівнянності та з метою усунення двомовних класів документів. .

Сантош [39] запропонував використання вмісту документа (тобто текстових полів) на основі інтелектуального аналізу графів. Тобто запит генерував графік залежно від вимог користувачів. Це простий і ефективний метод аналізу графіків для вилучення схожих шаблонів у документах і зміни графа запиту на модельні графіки, які використовуються, коли користувачі відсутні. Створено інтелектуальне рішення для використання інформації документів. Це характеризується простотою, зручністю використання, точністю, легкістю розробки та гнучкістю. Щоб зрозуміти моделі графів, не потрібна величезна колекція зображень документів. Крім того, оскільки навчання моделі займає в середньому менше 10 секунд для шаблону введення на клас, можна вносити зміни, поправки та заміни в шаблони введення. Показано, що середня ефективність використання інформації становить 86,64% як точність і 90,80% як відкликання. Однак запропонована техніка не змогла запропонувати всеохоплюючі та точні рішення для

шаблонів, які мають величезну колекцію полів у зигзагоподібному порядку через складність графа запиту.

Сірсат та ін. [23] запропонував дві методики видобутку тексту через онлайн-джерела. Перший метод стосувався знань, які потрібно показати безпосередньо в документах, які потрібно видобути. Текстовий аналіз і IE вважаються єдиними ефективними інструментами для виконання цієї техніки. Другий стосувався документів, які містять фактичні дані в неструктурованому форматі замість нефігуративних знань. IE може допомогти змінити неструктуровані дані, представлені в корпусі документів, на структуровані. Щоб виявити нефігуративні шаблони в витягнутих даних, можна використовувати алгоритми та методи інтелектуального аналізу даних.

Сонг і Кім [40] представив першу спробу застосувати підходи аналізу тексту до величезної колекції повних текстів статей для виявлення структури знань у цій галузі. Замість того, щоб залежати від даних про цитування, представлених у Web of Science, для бібліометричного дослідження використовувалися повнотекстові статті PubMed Central. Перш за все, це допомогло створити процедури видобутку тексту, щоб розробити спеціальну базу даних цитат після видобутку повного тексту. Висновки показали, що більшість опублікованих документів у галузі біоінформатики не цитувалися іншими. Крім того, спостерігалося постійне та лінійне зростання кількості публікацій протягом років видання. Результати показали, що більшість отриманих досліджень були натхненні американськими інститутами, а потім європейськими інститутами. Результати свідчать про те, що основна увага важливих тем була зосереджена на біологічних факторах. Однак, згідно з PageRank, топ-10 статей були дуже стурбовані обчислювальними факторами.

Для полегшення точного вилучення тексту з PDF-файлів дослідницьких статей, які можна використовувати в програмах для аналізу тексту, [41] була представлена система «Вилучення PDF-тексту з урахуванням макету» (LA-PDF Text) .]. Текстові блоки видобуваються з дослідницьких статей із

заповненням тексту у форматі PDF у цій системі, а потім система класифікує їх у логічні одиниці залежно від правил, які описують певні розділи. У системі LA-PDF Text зосереджено лише текстовий вміст дослідницьких статей. Ця система служить основою для нових експериментів у більш розвинутих методах вилучення мультимодального вмісту, наприклад зображень і графіків. Система проходить три фази: (1) Ідентифікація суміжних текстових блоків за допомогою обробки просторового макету з метою виявлення блоків суміжного тексту, (2) Класифікація текстових блоків у метафоричні категорії за допомогою методу на основі правил, і (3) об'єднання категоризованих текстових блоків разом шляхом їх точного впорядкування, що призводить до вилучення тексту з згрупованих за розділами блоків. Була проведена оцінка точності алгоритму виявлення блоку, використаного на кроці 2. Було також показано, що система може ідентифікувати та класифікувати їх за метафоричними категоріями з Recall = 0,89%, Precision = 0,96% і F = 0,91%. Крім того, точність тексту, отриманого за допомогою LA-PDF Text, порівнюється з текстом із підмножини відкритого доступу PubMed Central. Потім ця точність порівнюється з текстом, отриманим за допомогою системи PDF2Text. Це дві методи, які часто використовуються для вилучення тексту з PDF. точність тексту, отриманого за допомогою LA-PDF Text, порівнюється з текстом із підмножини відкритого доступу PubMed Central. Потім ця точність порівнюється з текстом, отриманим за допомогою системи PDF2Text. Це дві методи, які часто використовуються для вилучення тексту з PDF. точність тексту, отриманого за допомогою LA-PDF Text, порівнюється з текстом із підмножини відкритого доступу PubMed Central. Потім ця точність порівнюється з текстом, отриманим за допомогою системи PDF2Text. Це дві методи, які часто використовуються для вилучення тексту з PDF.

Муні та Бунеску [42] описав дві техніки використання вилучення інформації природною мовою для аналізу тексту. По-перше, загальні знання

можна отримати безпосередньо з тексту. Проект, у якому базу знань про 6580 взаємодій людського білка було отримано шляхом видобутку приблизно 750 000 тез Medline, у яких переглянуто як приклад цієї техніки. По-друге, структуровані дані можна видобувати через текстові документи або веб-сторінки. Щоб виявити закономірності в отриманих даних, можна застосувати традиційні методи KDD. Як приклад цієї методики було розглянуто виконану роботу над системою DiscoTEX та її застосування до описів книг Amazon, оголошень про роботу в галузі інформатики та резюме. Щоб виявити одиниці та зв'язки в тексті, дослідження в IE продовжують створювати ефективніші алгоритми. Цінні та важливі знання можна ефективно видобувати з масиву електронних документів і веб-сторінок, що постійно розвивається, використовуючи сучасні підходи в технології людської мови та комп'ютерній лінгвістиці, а також поєднуючи їх із сучасними методами, що використовуються в машинному навчанні та звичайних методах інтелектуального аналізу даних. IE має справу з визначенням певного набору релевантних елементів через документи природною мовою.

Щоб виявити теми, які повторюються в статтях текстового корпусу, інший метод TopCat (Topic Categories) був запропонований [22]. IE використовувався цією технікою, щоб виявити іменовані сутності в окремих статтях і охарактеризувати їх як набір елементів статті. Таким чином, шляхом розпізнавання частих наборів елементів, які зазвичай трапляються з іменованими сутностями, були вивчені проблеми інтелектуального аналізу даних або контексту бази даних. Для виявлення цих частих наборів елементів TopCat використовує техніку аналізу даних правила асоціації. Використовуючи техніку розбиття гіперграфа, TopCat додатково кластеризує названі сутності, виявляючи колекцію частих наборів елементів зі значним перекриттям. Для виявлення документів по темі було використано ІЧ-технологію. У цьому методі використовувалися різні технології, такі як IE для вилучення іменованих сутностей, інтелектуальний аналіз даних за правилами

асоціацій, кластеризація правил асоціацій, методи ІЧ. TopCat виявляє теми, які мають логічну точність з прийнятними ідентифікаторами. Каллан і Мітамура [43] представив нову техніку для виявлення іменованих об’єктів, яка називається KENE. Щоб зрозуміти правила вилучення, в ньому використовується методика, заснована на знаннях. Підхід генерування та перевірки використовується для вилучення іменованих сутностей зі структурованих документів.

З досліджуваної літератури ми можемо помітити, що існує обмеження у розгляді проблеми ІЕ з дослідницьких статей із використанням методів аналізу даних. Синергія між цими підходами (тобто ІЕ з методами інтелектуального аналізу даних) допомагає виявити різні цікаві текстові шаблони в отриманих статтях. Цей підхід може бути застосований до різноманітних дослідницьких тем, де в кожній темі можна створити широкий спектр моделей знань. Мобільне навчання (M-learning) стало одним із модних напрямків у вищій освіті [44 , 45 , 46 , 47 , 48 , 49 , 50].

1.3 Підходи інтелектуального аналізу тексту

Розвиток у сфері Інтернету, цифрових бібліотек, технічної документації, медичних даних полегшив доступ до більшої кількості текстових документів, які об’єднуються для створення корисних ресурсів даних [7] . Таким чином, це робить інтелектуальний аналіз тексту або виявлення знань із текстових баз даних складним завданням завдяки відповідності стандартам глибини природної мови, яка використовується в більшості доступних документів. Доступна текстова інформація у вигляді баз даних та онлайн-джерел [7 , 8 , 9] виникає питання про те, хто відповідає за перевірку даних і їх аналіз? З огляду на відповідну умову, неможливо проаналізувати та ефективно витягти корисну інформацію вручну. Існує потреба в застосуванні програмних рішень, які можуть використовувати автоматичні інструменти для

аналізу значної кількості текстового матеріалу, вилучення відповідних даних, аналізу відповідних даних та організації відповідної інформації. У зв'язку зі зростанням вимог щодо отримання знань із великої кількості текстових документів, доступних в Інтернеті, інтелектуальний аналіз тексту набуває значного значення в дослідженнях [10 , 11]. Загалом інтелектуальний аналіз тексту та інтелектуальний аналіз даних вважаються схожими один на одного, при цьому існує думка, що в обох концепціях для аналізу тексту можуть використовуватися однакові методи [4 , 12 , 13] і [3]. Однак обидва вони відрізняються в тому сенсі, що інтелектуальний аналіз даних передбачає структуровані дані, тоді як текст має справу з певними функціями та є відносно неструктурованим і зазвичай вимагає попередньої обробки. Крім того, видобуток тексту є взаємопов'язаною сферою з обробкою природної мови (NLP). NLP є однією з гарячих тем, яка стосується взаємозв'язку між величезною кількістю доступного неструктурованого тексту [14], окрім аналізу та інтерпретації людських мов [15 , 16].

Початковою точкою комп'ютерів для оцінки неструктурованих рукописів є використання вилучення інформації (IE). Програмне забезпечення IE розпізнає ключові фрази та зв'язки, включені в рукопис. Це виконується шляхом пошуку попередньо визначених порядків у тексті; цей прийом називається шаблоном. Текстові документи звичайною мовою складаються з інформації, яка не може бути використана для майнінгу. IE погоджується з документацією, вибираючи відповідні статті та асоціації між ними, щоб зробити їх більш доступними для додаткового керівництва [17 , 18]. На відміну від інформаційного пошуку, який має справу з тим, як розпізнати релевантні документи з колекції документів, IE надає структуровану інформацію, підготовлену для пост-обробки, яка є важливою для різних застосувань інструментів веб-видобутку та пошуку [19] . IE займається виявленням і вилученням важливої інформації з текстів природною мовою [18]. Він складається з розділення відповідних текстових частин, вилучення

запропонованих даних у таких частинах і перетворення даних у функціональну форму. Наразі можливе дробове вилучення з предметних текстів; хоча повний ІЕ з довільного тексту все ще є метою постійного дослідження [20].

За більшості умов з інформації, витягнутої з неструктурованого тексту, отримують лише конкретні дані, а не абстрактні знання. У такому випадку потрібно застосувати завдання інтелектуального аналізу тексту разом із додатковими техніками для отримання знань із наявних даних [21 , 22]. DiscoTEX (Discovery from Text EXtraction) є одним із основних підходів, що використовуються для видобутку тексту. Це передбачає використання ІЕ спочатку для збору структурованих даних із неструктурованого тексту, а потім використання традиційних інструментів виявлення знань із бази даних (KDD) для виявлення знань із цих даних. Цю структуру для аналізу тексту було представлено [21]. У цьому методі навчена система ІЕ використовується для перетворення неструктурованого тексту в більш структуровані дані. Ці дані потім піддаються інтелектуальному аналізу для створення значущих зв’язків. У випадку, коли інформація, отримана з корпусу документів, є у формі абстрактних знань замість конкретних даних, ІЕ, як правило, слугує «знанням, що відкриває» з тексту. Виявлення знань шляхом вилучення інформації, такої як ключові фрази або вилучення ключових слів із тексту, може бути використано для інших завдань інтелектуального аналізу тексту, тобто класифікації, кластеризації, узагальнення та виявлення теми [23] .

Текстовий аналіз зазвичай використовується для отримання швидких результатів [24]; він був підданий дослідженню в ряді областей застосування. На основі відповідних сфер застосування інтелектуальний аналіз тексту можна класифікувати як категоризація тексту, кластеризація тексту, вилучення правил асоціації та візуалізація тексту. Вони обговорюються в наступних підрозділах.

Кластеризація тексту базується на кластерній гіпотезі, яка передбачає, що релевантні документи повинні мати більше схожості один з одним, ніж нерелевантні [25]. Техніка кластеризації є надійною технікою, яка зазвичай використовується для аналізу великих обсягів даних, наприклад інтелектуального аналізу даних. Було доведено, що кластеризація тексту є одним із найефективніших інструментів, які використовуються для аналізу тем тексту [26]. Крім того, це полегшує метод тематичного аналізу, в якому іменовані сутності, які одночасно зустрічаються, групуються разом, з подальшим підпорядкуванням їх процесу кластеризації таким чином, що часті елементи розміщуються в наборах із застосуванням методу на основі гіперграфа [27] .]. Кожен набір іменованих сутностей представлений кластером, пов'язаним з однією з актуальних тем у корпусі. Процес відстеження теми в динамічних текстових даних зацікавив дослідників, які працюють над темою кластеризації тексту в цифровій сфері. У процес кластеризації документів включені різні методи та алгоритми, засновані на неконтрольованому управлінні документами. У процесі кластеризації числа, властивості та асоціації згрупованих наборів спочатку невідомі. Групування документів виконується шляхом класифікації їх за певною категорією, такою як медичні, фінансові та/або юридичні [28].

У дослідженні [29] стверджується, що метод аналізу правил асоціації (ARM) використовується для виявлення зв'язків у більшій групі змінних у наборі даних. ARM ідентифікує комбінації змінних значень, які, як правило, трапляються часто. Метод ARM в інтелектуальному аналізі даних також відомий як виявлення знань у базах даних; це подібно до кореляційного аналізу, який з'ясовує зв'язки між двома змінними. Вонг та ін. [30] за умови, що Правила асоціації для інтелектуального аналізу тексту в основному спрямовані на дослідження зв'язків між різними темами або фактичними поняттями, які використовуються для характеристики корпусу. Вони мають

намір виявити ключові правила асоціації щодо корпусу таким чином, щоб поява певних тем у статті могла відповідати появі іншої теми.

Підхід *k* -середнього ділить набір даних на *k* кластерів, де кожен кластер підлягає представленню середнього значення балів; називають центроїдом. Для застосування алгоритму використовується двоетапний повторюваний процес: (1) Присвоєння кожної точки найближчому центроїду. (2) Оцінка центроїдів для нещодавно розробленої групи. Процес завершується, коли центроїд кластера стає постійним. Алгоритм *k* -середніх має широке застосування завдяки прямому розпаралелюванню. Крім того, порядок відповідних даних не впливає на алгоритм *k* -середніх, який приписує їм числові характеристики. Необхідно зазначити максимальне значення *k*на початку процесу. Представлення кластера виконується алгоритмом *k* -medoid, який вибирає об'єкт, що примикає до центру кластера. Однак вибір *k* об’єктів виконується випадковим чином в алгоритмі. Вибрані предмети допомагають визначити відстань. Кластер формується на основі найближчого до *k* об’єкта , тоді як інші об’єкти набувають позиції *k* рекурсивно до досягнення необхідної якості кластера [28].

Візуалізація інформації розміщує чудові текстові бази у візуальній ієрархії або плані та пропонує можливості перегляду, а також загального пошуку. Ця техніка пропонує покращені та швидші всеосяжні знання, які допомагають нам видобувати величезні накопичення документів. Оператори можуть розрізняти кольори, асоціації та прогалини. Асортимент документів можна продемонструвати у вигляді структурованого макета з використанням індексування або векторної просторової моделі.

Джаяшанкар і Шрідаран [31] визначили хмари слів або хмари тегів як візуальне представлення слів для певного письмового вмісту, структурованого відповідно до його частоти. Хмара слів є одним із найбільш часто використовуваних методів представлення текстових даних у графічному вигляді; що робить його корисним для аналізу різних форм текстових даних,

таких як есе та короткі відповіді або письмові думки до опитування чи анкети [32]. Хмара слів, як правило, слугує попереднім етапом для поглибленого аналізу певного текстового матеріалу [33 , 34] .]. Наприклад, хмара слів допомагає знайти релевантність між даним текстом і необхідною інформацією. Тим не менш, метод має і певні недоліки. Одним із головних недоліків є те, що він не враховує лінгвістичні знання про слова та їх відповідний зв'язок із даним предметом, надаючи суто статистичне резюме для відокремлених слів. Як наслідок, у більшості систем хмари слів часто використовуються в статистичних цілях для підсумовування тексту, надаючи дуже мало або взагалі не надаючи засобів для кореляції даних. Вважається, що це може бути одна з найбільш впливових парадигм візуалізації для більшості умов аналізу.

1.4 Вибір перспективного шляху і постановка задачі дослідження

Актуальність теми полягає в тому, що в сучасному інформаційному суспільстві, де доступ до великого обсягу наукових публікацій є широко поширеним, знаходження та відбір наукових статей стає складною задачею. Люди з різних галузей залежать від доступу до актуальних досліджень та наукових статей для своєї роботи, навчання та розвитку.

Модуль підбору тематики наукових статей на основі NLP має великий потенціал у поліпшенні цього процесу. За допомогою природньо-мовних технологій (NLP), модуль може автоматично аналізувати текст статей, виявляти їхню тематику та порівнювати з запитами користувачів. Це дозволяє забезпечити більш точний та релевантний підбір статей, що відповідають потребам та інтересам користувачів.

Крім того, зростання обсягу наукової інформації ставить завдання перед науковцями та дослідниками з ефективним відбором та аналізом наукових публікацій для своїх досліджень. Модуль на основі NLP може виявити

схожість та пов'язаність між різними статтями, що сприятиме ефективному пошуку та використанню відповідної літератури.

Таким чином, модуль підбору тематики наукових статей на основі NLP має велике значення для забезпечення доступу до релевантної та актуальної наукової інформації, поліпшення якості досліджень та сприяння розвитку науково-дослідницької діяльності.

Метою бакалаврського проекту є розробка та реалізація модуля підбору тематики наукових статей на основі NLP.

Для досягнення поставленої мети було необхідно провести наступні теоретичні розробки та виконати послідовні завдання:

1. Проаналізувати передумов для інтелектуального аналізу тексту та існуючі рішення.
2. Розробити структуру обробки інтелектуального аналізу тексту.
3. Розробити алгоритм обробки та розподілу набору даних для текстового аналізу.
4. Розробити алгоритм TF-IDF для підбору тематики наукових статей та алгоритм тренування моделі Doc2Vec.
5. Розробити алгоритм кластеризації документів для підбору тематики наукових статей.
6. Провести попередню обробку даних для інтелектуального аналізу тексту.
7. Натренувати моделі Doc2Vec для підбору тематики наукових статей.
8. Провести кластеризацію документів для підбору тематики наукових статей

Завдання 1 розглянуто в параграфах 1.1 - 1.4 відповідно, а виконання завдань 2-8 представлені у параграфах 2.1-3.4.

2 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ПІДБОРУ ТЕМАТИКИ НАУКОВИХ СТАТЕЙ

2.1 Структура обробки інтелектуального аналізу тексту

Розроблено індивідуальну структуру, яка базується на розробленій структурі, запропонованій [51], див. рис. 2.1.

Інтелектуальний аналіз тексту включає три етапи: попередня обробка тексту, операції аналізу тексту та постобробка.

Попередня обробка тексту включає в себе наступні завдання: відбір даних, класифікація, виділення ознак і нормалізація тексту, тобто перетворення документів у проміжну форму для забезпечення сумісності з різними інструментами видобутку.

Другий крок стосується різних методів інтелектуального аналізу тексту, таких як кластеризація, виявлення правил асоціації, візуалізація та частота термінів.

Під час третього кроку вносяться зміни та зміни в дані за допомогою функцій аналізу тексту, таких як оцінка та вибір знань, аналіз та візуалізація знань.

Модуль підбору тематики наукових статей на основі обробки природної мови (NLP) включає три основні етапи: попередню обробку тексту, операції аналізу тексту та постобробку.

1. Попередня обробка тексту: Перший етап полягає в обробці вхідних наукових статей перед подальшим аналізом. Цей етап може включати такі завдання, як відбір даних, класифікація, виділення ознак та нормалізація тексту. Відбір даних може включати вибір конкретних статей на основі певних критеріїв, таких як автор, журнал або ключові слова. Класифікація може використовуватися для впорядкування статей за певними категоріями або темами. Виділення ознак може включати

виявлення ключових слів, термінів або фраз, які є важливими для подальшого аналізу. Нормалізація тексту зазвичай включає приведення тексту до єдиного формату, видалення непотрібних символів, лематизацію та стемінг.

2. Операції аналізу тексту: Другий етап включає використання різних методів інтелектуального аналізу тексту для виявлення тематики наукових статей. Це можуть бути методи, такі як кластеризація, виявлення правил асоціації, візуалізація та частотний аналіз термінів. Кластеризація допомагає групувати статті за схожістю їх змісту. Виявлення правил асоціації дозволяє виявити зв'язки між термінами або концепціями в статтях. Візуалізація може бути використана для графічного представлення зв'язків між темами статей.

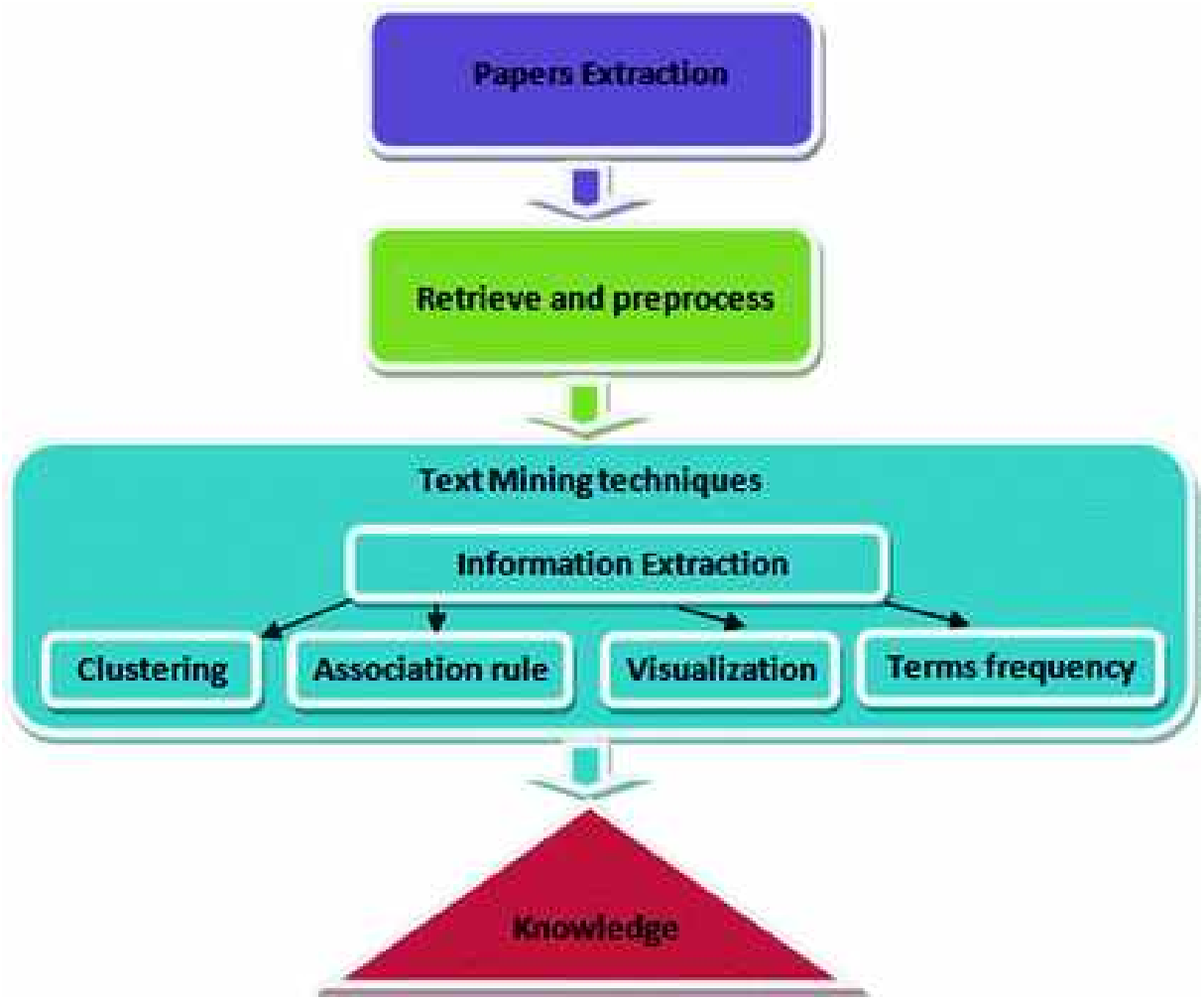


Рисунок 2.1 - Фреймворк підбору тематики наукових статей на основі NLP

3. Постобробка: Під час третього етапу вносяться зміни та покращення в дані за допомогою функцій аналізу тексту. Це може включати оцінку та вибір знань, аналіз та візуалізацію знань. Оцінка та вибір знань можуть використовуватися для визначення важливості та релевантності статей у контексті певної тематики. Аналіз та візуалізація знань можуть допомогти зрозуміти структуру та зв'язки між темами статей, що допомагає при подальшому виборі тематики.

Загалом, модуль підбору тематики наукових статей на основі NLP використовує різноманітні методи та техніки для аналізу тексту з метою визначення тематичної спрямованості статей. Це допомагає збільшити ефективність і точність підбору статей для конкретної задачі або дослідження.

2.2 Алгоритм обробки та розподілі набору даних для подальшого текстового аналізу

Обробка та розподіл набору даних - це процеси, які включають в себе ряд дій, спрямованих на підготовку та організацію набору даних для подальшого використання. Ці процеси можуть бути важливими перед виконанням аналітичних завдань або побудовою моделей машинного навчання.

Обробка набору даних включає наступні дії:

1. Очищення даних: В цьому етапі виявляються та видаляються аномальні, неповні або некоректні дані. Це може включати видалення дублікатів, обробку відсутніх значень, виправлення помилок та інші операції, що забезпечують якість даних.

2. Вибірка даних: Часто набори даних можуть бути дуже великими, і не всі дані є необхідними для конкретної задачі. У таких випадках може бути виконана вибірка даних, що полягає в обранні підмножини даних, які мають бути оброблені та використані.
3. Нормалізація та масштабування: Деякі алгоритми машинного навчання вимагають, щоб всі ознаки були на одному масштабі, тому нормалізація та масштабування даних можуть бути проведені. Це дозволяє забезпечити рівні умови для аналізу та моделювання.

Алгоритм, що зосереджується на обробці та розподілі набору даних для подальшого використання аналізу тексту, може включати такі кроки:

1. Отримання набору даних: Алгоритм починається з отримання необхідного набору даних, який може бути зібраний з різних джерел або створений власноруч. Цей набір даних може містити інформацію про користувачів, продукти, текстові документи або інші об'єкти, залежно від конкретної задачі.
2. Ресемплінг даних: Часом може бути необхідно зменшити розмір набору даних або збалансувати його склад, особливо якщо вихідний набір містить нерівномірно розподілені дані або дуже велику кількість записів. Ресемплінг даних може включати вибір підмножини записів або методи балансування класів.
3. Створення DataFrame: Після отримання набору даних, він зазвичай перетворюється у структуровану форму, таку як DataFrame, яка зручна для подальшої обробки та аналізу. DataFrame - це таблицеподібна структура даних, що складається з рядків та стовпців, де кожен стовпець представляє певну ознаку або атрибут.
4. Фільтрування ознак: Залежно від конкретної задачі, може бути необхідно вибрати лише певні ознаки або атрибути з набору даних для подальшого використання. Це може зменшити обсяг даних і сконцентрувати увагу на найважливіших аспектах.

5. Створення корпусу тексту: Якщо алгоритм спрямований на текстовий аналіз, то може бути необхідно створити корпус текст у. Корпус тексту представляє собою колекцію текстових документів, яку можна використовувати для подальшого аналізу тексту, такого як виявлення тем, класифікація, розпізнавання іменованих сутностей тощо. Для створення корпусу тексту, алгоритм може об'єднати всі текстові документи з набору даних в одну структуру, яка дозволяє легко отримувати доступ до тексту кожного документа.

Описані кроки вступу допомагають алгоритму підготувати дані для подальшого використання в рекомендаційних системах або текстовому аналізі. Це включає оптимізацію розміру набору даних, структуризацію його в зручний формат та вибір лише необхідних ознак або атрибутів. Крім того, створення корпусу тексту дозволяє алгоритму ефективно працювати з текстовими даними і виконувати різні завдання аналізу.

Алгоритм зосереджується на обробці та розподілі набору даних для подальшого використання текстового аналізу для поставленої задачі в дипломному проекті. Ось стислий опис кожного кроку та блок-схемою (див.дод.А):

Крок 1. Ресемплінг даних: Використовуючи функцію split_records, записи в data_records перетасовуються та розподіляються на дві частини: profile_records (перші 500 записів) та recommend_records (всі інші записи). Це може бути корисним для створення профілю користувача (на основі profile_records) та для генерації рекомендацій (на основі recommend_records).

Крок 2. Створення DataFrame: Використовуючи функцію get_dataframe, recommend_records (список словників) перетворюється на DataFrame pandas. Це полегшує обробку та аналіз даних.

Крок 3. Фільтрування ознак: Використовуючи функцію filter_features, recommend_df фільтрується так, що залишаються лише корисні ознаки

('title', 'categories', 'abstract', 'update_date'). Це забезпечує, що використовуються лише релевантні дані, що можуть покращити якість рекомендацій.

Крок 4. Створення корпусу тексту: Нарешті, з 'abstract' колонки recommend_df беруться перші 10000 записів та зберігаються в змінній corpus. Цей корпус може бути використаний для текстового аналізу, побудови моделі тематичного моделювання, або як вхідні дані для моделі машинного навчання.

Цей алгоритм може бути використаний як перший крок у більш складному процесі, такому як побудова рекомендаційної системи або текстового аналізатора.

Алгоритм обробки та розподілу набору даних для подальшого текстового аналізу є важливою складовою Модуля підбору тематики наукових статей на основі NLP. Цей алгоритм забезпечує оптимальну підготовку даних перед застосуванням методів обробки природної мови (NLP) для виявлення тематики наукових статей.

Вступний процес алгоритму включає ресемплінг даних, створення DataFrame, фільтрування ознак та створення корпусу тексту. Ці кроки дозволяють підготувати дані із вихідного набору, збалансувати його розмір, структурувати текстові дані та відібрати важливі ознаки.

Цей алгоритм дозволяє підготувати дані перед подальшим застосуванням NLP-методів для виявлення тематики наукових статей. Завдяки обробці та розподілу даних, алгоритм забезпечує належну якість даних, оптимальний розмір набору даних та структурованість текстових документів. Це сприяє ефективному використанню NLP-методів для підбору тематики наукових статей.

Таким чином, алгоритм обробки та розподілу набору даних відіграє важливу роль у побудові Модуля підбору тематики наукових статей на основі

NLP, допомагаючи забезпечити належну якість та підготовку даних перед використанням методів NLP для аналізу тексту.

2.3 Алгоритм TF-IDF для підбору тематики наукових статей

Даний алгоритм використовує метод званий TF-IDF (частота терміна-обернена частота документа) та косинусна подібність для знаходження документів, які найбільш схожі на заданий документ у корпусі документів. Ось його основні кроки та представлено блок-схемою (рис.2.2):

Крок 1. Використовуючи TfidfVectorizer з бібліотеки sklearn, алгоритм перетворює корпус тексту в матрицю TF-IDF. Кожен рядок в цій матриці відповідає документу, а кожний стовпець відповідає терміну (слову). Значення в матриці відображають важливість кожного терміна в кожному документі.

Крок 2. Потім алгоритм розраховує косинусну подібність між усіма парами документів, використовуючи матрицю TF-IDF. Косинусна подібність - це метрика, яка вимірює косинус кута між двома векторами. Вона використовується для вимірювання того, наскільки схожі між собою два документи.

Крок 3. Для отримання рекомендацій на основі заданого документа, алгоритм використовує функцію get_recommendations. Вона приймає абстракт заданого документа, матрицю косинусної подібності та індекси, і повертає 10 найбільш подібних документів.

Крок 4. У цьому діалозі було демонстровано, як алгоритм може бути використаний для отримання рекомендацій на основі різних документів. Наприклад, було вибрано документи з індексами 1, 500 та іншими для отримання рекомендацій.

Цей алгоритм може бути особливо корисним для рекомендацій в наукових або дослідницьких базах даних, де корисно знати, які інші документи є найбільш схожими на даний документ.

Алгоритм, розглянутий у цьому діалозі, використовує методи обробки природної мови (NLP) для рекомендації наукових статей, що мають схожу тематику. Він використовує техніку TF-IDF для визначення важливості слів в контексті корпусу тексту і косинусну подібність для вимірювання подібності між різними документами.

Цей алгоритм може бути використаний як основа для модулю підбору тематики наукових статей, дозволяючи користувачам швидко і ефективно знайти статті, які найбільш відповідають їх інтересам або поточній області досліджень.

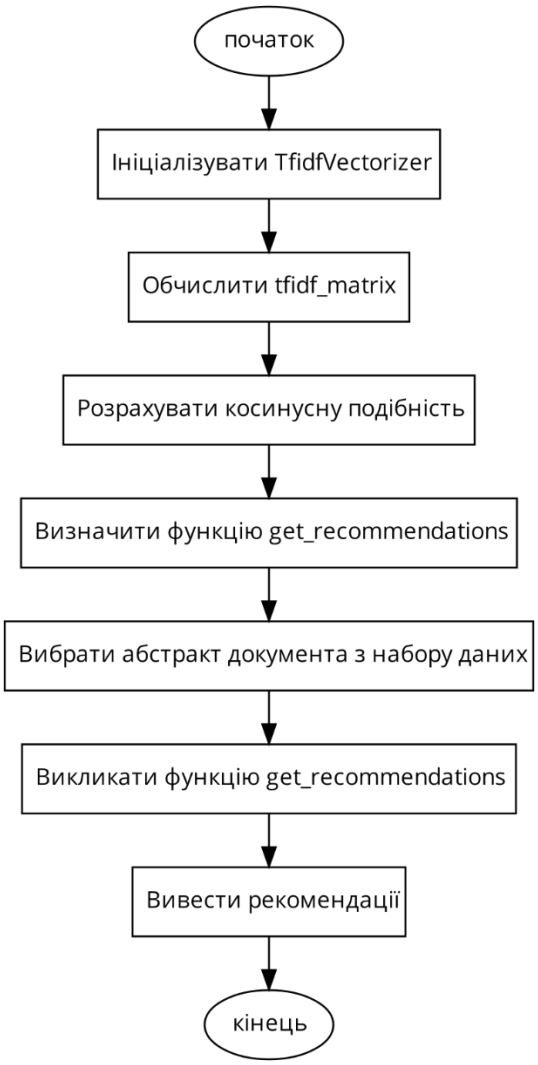


Рисунок 2.2 - Алгоритм TF-IDF для підбору тематики наукових статей

2.4 Алгоритм тренування моделі Doc2Vec для підбору тематики наукових статей

Алгоритм, який буде представлено, стосується області обробки природної мови (NLP) і використовує Doc2Vec, один з алгоритмів векторного представлення слова, розроблених командою Google. Цей алгоритм є частиною більшої системи для підбору тематики наукових статей на основі їхнього абстракту.

Doc2Vec створює вектори для цілих документів, замість окремих слів, що дозволяє виявити схожість між різними документами. Це корисно для

численних застосувань, включаючи рекомендації, класифікацію та кластеризацію статей.

Алгоритм включає підготовку та обробку даних, тренування моделі Doc2Vec, використання моделі для знаходження схожих документів і аналіз результатів. Наступні розділи подробиці розкажуть про кожен з цих кроків.

Ось алгоритм, який тренує модель Doc2Vec для знаходження схожих документів на основі вхідного документа представлений наступними кроками та блок-схемою (див.дод.Б):

Крок 1. Підготовка даних:

1.1. Створити новий стовпець abstract_len в DataFrame recommend_df, який обчислює кількість слів в кожному абстракті.

1.2. Завантажити NLTK і використати її для токенізації кожного документа в наборі даних corpus.

1.3. Призначити кожному документу унікальний ідентифікатор.

Крок 2. Створення та тренування моделі Doc2Vec:

2.1. Створити модель Doc2Vec.

2.2. Побудувати словник для моделі на основі підготовлених даних.

2.3. Тренувати модель протягом декількох епох, зменшуючи швидкість навчання після кожної епохи.

2.4. Зберегти модель для подальшого використання.

Крок 3. Використання моделі для знаходження схожих документів:

3.1. Завантажити збережену модель.

3.2. Токенізувати вхідний документ (в даному випадку, документ з індексом 700 в DataFrame recommend_df).

3.3. Використати модель для знаходження документів, що найбільш схожі на вхідний документ.

3.4. Вивести найбільш схожі документи.

Крок 4. Аналіз результатів:

4.1. Вивести категорію та абстракт вхідного документа.

4.2. Вивести категорію та абстракт найбільш схожого документа.

4.3. Порівняти категорії та абстракти обох документів, щоб перевірити, чи документи дійсно схожі за їхнім вмістом та темою.

Алгоритм використовує модель NLP Doc2Vec для підбору тематики наукових статей. Він токенізує тексти, тренує модель, знаходить найбільш схожі статті та аналізує результати. Виділені статті можуть використовуватись для рекомендацій, класифікації або кластеризації.

Алгоритм, розроблений для модуля підбору тематики наукових статей на основі NLP, успішно використовує технологію Doc2Vec для знаходження схожості між різними науковими статтями. Через тренування моделі на корпусі текстів, алгоритм здатний виявляти семантичні зв'язки між документами, що може бути використано для рекомендацій, класифікації та кластеризації статей. Цей підхід покращує здатність системи підбору тематики статей, роблячи його більш точним та ефективним.

2.5 Алгоритм кластеризації документа для підбору тематики наукових статей

Алгоритм, який будемо розглядати, використовує техніки машинного навчання та обробки природної мови для групування текстових даних. Це особливо корисно для великих наборів даних, де ручне сортування або категоризація може бути непрактичним або неможливим.

Застосуваннями цього алгоритму можуть бути, наприклад, групування новинних статей за темами, сортування відгуків користувачів за схожістю вмісту, або структуризація наукових статей за досліджуваними питаннями.

Алгоритм використовує перетворення TF-IDF для кількісного представлення тексту, аналіз головних компонент (PCA) для зменшення розмірності даних, метод ліктя для визначення оптимального числа кластерів, K-середніх для кластеризації, та t-SNE для візуалізації результатів у двовимірному просторі.

Мета цього алгоритму - знайти структуру в даних, використовуючи незалежність від людини методи машинного навчання, що дозволяє нам зрозуміти, як дані можуть бути згруповані або організовані на основі вмісту тексту.

Отже, алгоритм можна описати так та представити у вигляді блок-схеми (рис. 2.3):

Крок 1. **Підготовка даних:** Завантажте та підготуйте свій набір даних. У цьому випадку, ви використовуєте корпус текстових даних та набір даних **recommend_df**, який містить назви документів.

Крок 2. **Перетворення даних:** Використовуйте TF-IDF для перетворення корпусу текстових даних у числову матрицю **tfidf_matrix**. TF-IDF знаходить важливість кожного слова в документі в контексті всього корпусу.

Крок 3. **Зменшення розмірності:** Використовуйте PCA з **n_components=0.95** для зменшення розмірності **tfidf_matrix** до матриці **df_reduced**, яка зберігає 95% варіативності оригінальних даних.

Крок 4. **Визначення оптимального числа кластерів:** Використовуйте метод ліктя для визначення оптимального числа кластерів для KMeans, перебираючи різне число кластерів і вираховуючи суму квадратів відстаней (SSE) для кожного числа кластерів.

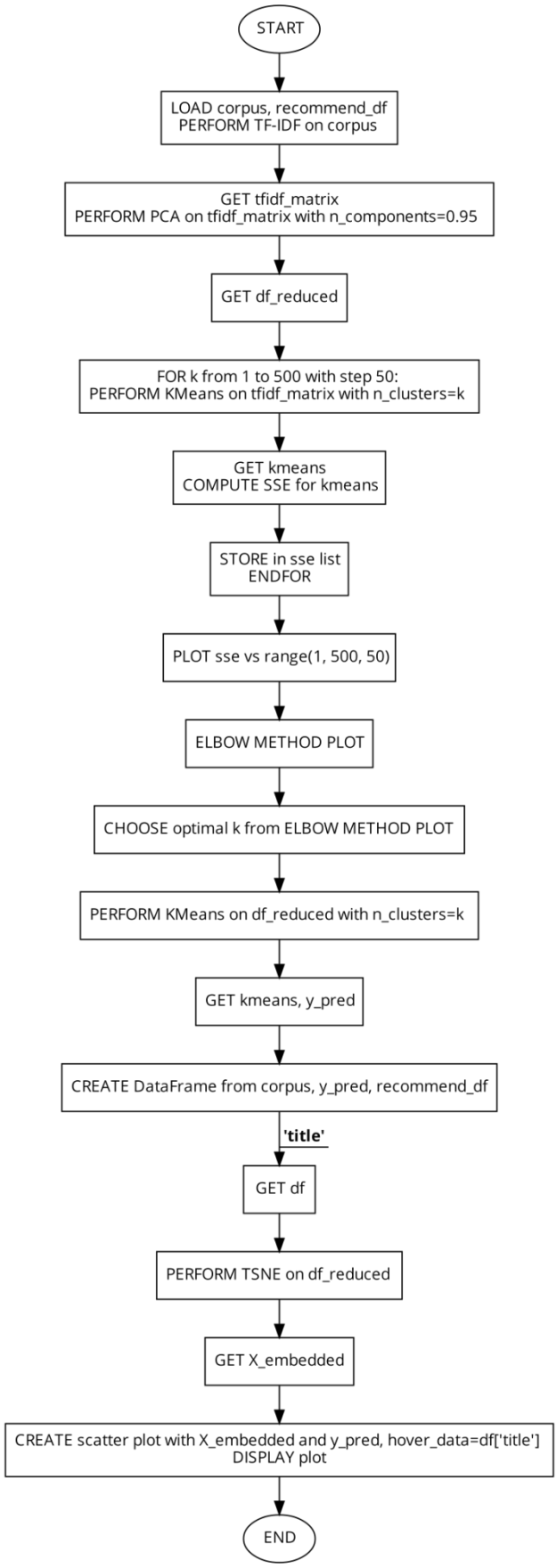


Рисунок 2.3 - Алгоритм кластеризації документа для підбору тематики наукових статей

Крок 5. **Кластеризація:** Використовуйте KMeans з оптимальним числом кластерів, щоб розділити **df_reduced** на кластери. Отримайте мітки кластера для кожного документа.

Крок 6. **Створення DataFrame:** Об'єднайте корпус, мітки кластера та назви документів у новий DataFrame для подальшого аналізу.

Крок 7. **Додаткове зменшення розмірності та візуалізація:** Використовуйте t-SNE для перетворення **df_reduced** до двовимірного простору, що дозволяє візуалізацію даних на графіку.

Крок 8. **Візуалізація результатів:** Використовуйте бібліотеку Plotly для створення інтерактивного графіка розсіювання, де кожна точка представляє документ, а її колір відповідає мітці кластера. При наведенні курсора на точку відображається назва відповідного документа.

Використання цього алгоритму дозволяє згрупувати великі набори текстових даних на основі їхнього вмісту, що може бути корисним для розуміння загальних тем або структури в наборі даних. Додатково, візуалізація допомагає краще зрозуміти, як документи групуються, та дозволяє інтерактивно досліджувати результати кластеризації.

Отже, представлений алгоритм кластеризації документа базується на NLP і машинному навчанні. Він використовує TF-IDF, PCA та KMeans для групування статей за темами, а потім використовує t-SNE для візуалізації цих груп в двовимірному просторі.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ ПІДБОРУ ТЕМАТИКИ НАУКОВИХ СТАТЕЙ

3.1 Попередня обробка даних

Для реалізації поставлених задач дипломного проекту використаємо набір даних ArXiv, доступний на Kaggle,[69] який є репозиторієм більш ніж 1,7 мільйона статей з різних наукових галузей. Цей набір даних містить метадані про кожну статтю та надає інформацію, таку як заголовок, автори, категорії, анотації, повний текст у форматі PDF та багато іншого.

Набір даних ArXiv на Kaggle надає лише метадані у форматі JSON, але повний набір PDF-файлів статей доступний для завантаження через Google Cloud Storage або Google API.

Цей набір даних відіграє важливу роль у полегшенні доступу до наукової літератури та може бути використаний для багатьох цікавих досліджень та аналітичних завдань.

Розглянемо більш детальніше характеристики набору даних, але спершу опишемо підключення цього набору даних.

Цей набір даних має 249500 записів, розподілених у 4 колонки. Всі типи даних у цих колонках є об'єктами. Ось деталі кожної колонки:

1. "title": Колонка з ненульовими значеннями, яка містить заголовки (назви) для кожного запису.
2. "categories": Колонка, яка також містить лише ненульові значення. Ця колонка містить категорії, до яких належить кожний запис.
3. "abstract": Ще одна колонка з ненульовими значеннями. Містить короткий опис або резюме для кожного запису.
4. "update_date": Колонка, яка містить дати останнього оновлення для кожного запису. Всі її значення також є ненульовими.

Загальний обсяг пам'яті, який займає цей DataFrame, становить близько 1.1+ гігабайти.

```
#Resampling method to fetch records for generating user_profile and recommendation algorithm
def split_records(data_records, profile_capacity=100, random_state=42):
    np.random.seed(random_state)
    np.random.shuffle(data_records)
    profile_records, recommend_records = data_records[:profile_capacity], data_records[profile_capacity:]
    return profile_records, recommend_records

#Splitting the fetched records into profile and recommendation records
profile_records, recommend_records = split_records(data_records, profile_capacity=500)
```

Далі використали метод ресемплінгу для розподілу записів даних на дві частини: profile_records і recommend_records. Це робиться з метою створення профілю користувача та використання алгоритму рекомендацій.

Спочатку ініціалізується генератор випадкових чисел з фіксованим random_state=42, щоб гарантувати повторюваність результатів. Потім ви перемішуєте записи в data_records.

Функція split_records розділяє data_records на дві частини: profile_records та recommend_records. Кількість записів в profile_records визначається параметром profile_capacity, який ви встановили рівним 500.

- profile_records: Це перші 500 записів з data_records після їх перемішування. Ці записи будуть використані для формування профілю користувача.
- recommend_records: Це усі записи, що залишилися після відбору 500 для profile_records. Ці записи будуть використані для роботи алгоритму рекомендацій.

Зверніть увагу, що кількість записів в recommend_records залежить від загальної кількості записів в data_records і може бути різною.

```
def get_dataframe(list_of_dicts, columns=None):
    data = pd.DataFrame(list_of_dicts)
    if columns:
        data.columns = columns
    return data

#Generating dataframes for profile and recommend records
recommend_df = get_dataframe(recommend_records)
```

Далі використаємо допоміжну функцію get_dataframe для перетворення списку словників (recommend_records) у DataFrame pandas. Функція

get_dataframe приймає список словників та опціональний список назв колонок як вхідні параметри. Вона створює DataFrame на основі вхідного списку словників, використовуючи pd.DataFrame(). Якщо список назв колонок передано, функція змінює назви колонок DataFrame на задані.

В результаті, recommend_df - це DataFrame, що створений на основі recommend_records. Цей DataFrame містить ті ж дані, що і recommend_records, але представлені в структурованій формі таблиці, що є зручнішим для аналізу даних і машинного навчання. Назви колонок у recommend_df відповідають ключам у словниках в recommend_records, якщо не було задано інших назв колонок.

```
#Utility method to filter out certain features which are of use
def filter_features(data, features):
    return data[features]

#Filtering the profile and recommendation dataframes for useful features
useful_features = ['title', 'categories', 'abstract', 'update_date']
recommend_df = filter_features(recommend_df, useful_features)
```

За допомогою допоміжної функції filter_features, відфільтровано DataFrame recommend_df, залишаючи лише корисні ознаки. Функція filter_features приймає DataFrame і список ознак як вхідні параметри і повертає DataFrame, який містить лише зазначені ознаки. Визначили useful_features як список, який містить назви чотирьох ознак: 'title', 'categories', 'abstract', 'update_date'. Тоді застосували filter_features до recommend_df, передавши useful_features як список ознак для відбору. Результатом є новий DataFrame recommend_df, який містить лише вказані ознаки.

Тепер recommend_df містить лише колонки 'title', 'categories', 'abstract' і 'update_date'. Це означає, що будь-які інші колонки, які були в початковому DataFrame, були видалені.

Далі вибрали перші 10000 значень з колонки 'abstract' DataFrame **recommend_df** і присвоїли їх змінній **corpus**. **corpus** тепер є pandas Series, який містить перші 10000 абстрактних описів з **recommend_df**.

3.2 Застосування TF-IDF

Векторизація тексту - це процес перетворення тексту в числові вектори, які можуть бути аналізовані та оброблені алгоритмами машинного навчання. Одним з найпопулярніших методів векторизації тексту є TF-IDF (Term Frequency-Inverse Document Frequency), який вимірює важливість кожного слова в документі в контексті всього корпусу документів.

У наступному коді ми розглянемо, як використовувати TF-IDF для векторизації тексту за допомогою бібліотеки Python sklearn.

```
start = time.time()
# Initialize an instance of tf-idf Vectorizer
tfidf_vectorizer = TfidfVectorizer()

# Generate the tf-idf vectors for the corpus
tfidf_matrix = tfidf_vectorizer.fit_transform(corpus)
```

Цей код використовує sklearn бібліотеку для векторизації тексту за допомогою TF-IDF (Term Frequency-Inverse Document Frequency).

1. start = time.time() - цей рядок ініціалізує час початку виконання скрипта. time.time() повертає поточний час у секундах з моменту початку епохи (1 січня 1970 року). Це може бути корисним для вимірювання тривалості виконання скрипта.
2. tfidf_vectorizer = TfidfVectorizer() - цей рядок створює екземпляр класу TfidfVectorizer. Цей векторизатор використовується для перетворення колекції сирого документів на матрицю TF-IDF ознак.
3. tfidf_matrix = tfidf_vectorizer.fit_transform(corpus) - тут fit_transform викликається на об'єкті tfidf_vectorizer, щоб вивчити словник (idf - Inverse Document Frequency) від нашого corpus та перетворити corpus на TF-IDF представлення.

corpus - це список документів, які потрібно перетворити. Кожен документ - це рядок тексту.

TF-IDF - це числова статистика, яка має на меті відобразити, наскільки важливе слово для документа в колекції документів. Це один з найпопулярніших способів векторизації тексту.

```
# compute and print the cosine similarity matrix
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
print(cosine_sim)
print("Time taken: %s seconds" % (time.time() - start))
```

Цей код розраховує та виводить матрицю косинусної подібності для нашої матриці TF-IDF, а також вимірює час виконання скрипта.

1. cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix) - цей рядок використовує функцію cosine_similarity з sklearn для розрахунку матриці косинусної подібності для нашої матриці TF-IDF. Косинусна подібність - це міра подібності між двома векторами, яка розраховує косинус кута між ними. Це широко використовується в системах рекомендацій та інших задачах, де потрібно знайти подібність між документами або користувачами.
2. print(cosine_sim) - цей рядок просто виводить матрицю косинусної подібності.
3. print("Time taken: %s seconds" % (time.time() - start)) - цей рядок вимірює тривалість виконання скрипта. time.time() - start розраховує час, що пройшов від початку виконання скрипта (коли була встановлена змінна start) до поточного моменту. Результат виводиться в секундах.

Результат (Рис.3.1) складається з двох частин: матриці косинусної подібності та часу виконання. Матриця косинусної подібності: Це двовимірна матриця, де кожен рядок та стовпець відповідають документу в корпусі. Кожне значення в матриці відображає косинусну подібність між двома документами. Значення варіюють від 0 до 1, де 0 означає, що документи повністю відрізняються, а 1 - що вони ідентичні. Наприклад, перше значення в кожному рядку (та відповідно в кожному стовпці) завжди дорівнює 1, оскільки це подібність документа з самим собою. Час виконання: Час виконання скрипта складає 8.53 секунди. Це включає час, потрібний для

ініціалізації векторизатора TF-IDF, перетворення корпусу в матрицю TF-IDF, розрахунку матриці косинусної подібності та виводу результатів. Залежно від розміру корпусу та потужності комп'ютера, час виконання може відрізнятися.

```
[[1.         0.04873471 0.07311138 ... 0.11822105 0.04931351 0.06621995]
 [0.04873471 1.         0.05549898 ... 0.08165059 0.04139559 0.0333875 ]
 [0.07311138 0.05549898 1.         ... 0.07745652 0.04586794 0.04374514]
 ...
 [0.11822105 0.08165059 0.07745652 ... 1.         0.05688083 0.04185297]
 [0.04931351 0.04139559 0.04586794 ... 0.05688083 1.         0.03339095]
 [0.06621995 0.0333875  0.04374514 ... 0.04185297 0.03339095 1.        ]]
Time taken: 8.526836633682251 seconds
```

Рисунок 3.1 - Матриця косинусної подібності та часу виконання

```
# Record start time
start = time.time()

# Compute cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

# Print cosine similarity matrix
print(cosine_sim)

# Print time taken
print("Time taken: %s seconds" % (time.time() - start))
```

Цей код дуже схожий на попередній, але він використовує linear_kernel замість cosine_similarity для розрахунку матриці подібності.

1. start = time.time() - цей рядок ініціалізує час початку виконання скрипта. time.time() повертає поточний час у секундах від початку епохи (1 січня 1970 року). Це може бути корисним для вимірювання тривалості виконання скрипта.
2. cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix) - тут linear_kernel викликається на tfidf_matrix, щоб розрахувати матрицю подібності. Лінійне ядро використовується для розрахунку внутрішнього (скалярного) добутку двох векторів, що в даному випадку еквівалентно косинусній подібності.
3. print(cosine_sim) - цей рядок виводить матрицю подібності.
4. print("Time taken: %s seconds" % (time.time() - start)) - цей рядок вимірює тривалість виконання скрипта. time.time() - start розраховує час, що

пройшов від початку виконання скрипта (коли була встановлена змінна start) до поточного моменту. Результат виводиться в секундах.

Результат (Рис.3.2) складається з двох частин: матриці подібності, що визначена за допомогою лінійного ядра, та часу виконання.

1. Матриця подібності: Це двовимірна матриця, де кожен рядок та стовпець відповідають документу в корпусі. Кожне значення в матриці відображає подібність між двома документами, розраховану за допомогою лінійного ядра (що в даному випадку еквівалентно косинусній подібності). Значення варіюють від 0 до 1, де 0 означає, що документи повністю відрізняються, а 1 - що вони ідентичні. Наприклад, перше значення в кожному рядку (та відповідно в кожному стовпці) завжди дорівнює 1, оскільки це подібність документа з самим собою.
2. Час виконання: Час виконання скрипта складає 7.07 секунди. Це включає час, потрібний для ініціалізації векторизатора TF-IDF, перетворення корпусу в матрицю TF-IDF, розрахунку матриці подібності за допомогою лінійного ядра та виводу результатів. В цьому випадку виконання коду зайняло менше часу, ніж в попередньому прикладі з **cosine_similarity**, але конкретні часи можуть варіюватися залежно від розміру корпусу та потужності комп'ютера.

```
[[1.         0.04873471 0.07311138 ... 0.11822105 0.04931351 0.06621995]
 [0.04873471 1.         0.05549898 ... 0.08165059 0.04139559 0.0333875 ]
 [0.07311138 0.05549898 1.         ... 0.07745652 0.04586794 0.04374514]
 ...
 [0.11822105 0.08165059 0.07745652 ... 1.         0.05688083 0.04185297]
 [0.04931351 0.04139559 0.04586794 ... 0.05688083 1.         0.03339095]
 [0.06621995 0.0333875  0.04374514 ... 0.04185297 0.03339095 1.        ]]
Time taken: 7.07038426399231 seconds
```

Рисунок 3.2. Матриця подібності, що визначена за допомогою лінійного ядра, та часу виконання

```python
indices = pd.Series(recommend_df.index, index=recommend_df['abstract']).drop_duplicates()

def get_recommendations(abstract, cosine_sim, indices):
    # Get the index of the movie that matches the title
    idx = indices[abstract]
    # Get the pairwsie similarity scores
    sim_scores = list(enumerate(cosine_sim[idx]))
    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    # Get the scores for 10 most similar movies
    sim_scores = sim_scores[1:11]
    # Get the movie indices
    paper_indices = [i[0] for i in sim_scores]
    # Return the top 10 most similar movies
    return recommend_df['abstract'].iloc[paper_indices]
```

Цей код визначає функцію get_recommendations, яка приймає тезу наукової роботи (абстракт), матрицю косинусної подібності та індекси, і повертає 10 найбільш подібних наукових робіт на основі їхніх тез. Ось більш детальний аналіз кожного кроку:

1. idx = indices[abstract]: цей рядок знаходить індекс наукової роботи, що відповідає заданій тезі.
2. sim_scores = list(enumerate(cosine_sim[idx])): тут для кожної наукової роботи в корпусі розраховується косинусна подібність з заданою роботою. Результат представляє собою список кортежів, де перший елемент - це індекс наукової роботи, а другий - її косинусна подібність до заданої роботи.
3. sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True): рядок сортує наукові роботи на основі їхніх косинусних подібностей до заданої роботи в порядку зменшення.
4. sim_scores = sim_scores[1:11]: цей рядок забирає перші 10 наукових робіт з відсортованого списку. Індекс 0 відкидається, оскільки він відповідає заданій науковій роботі (косинусна подібність документа з самим собою завжди дорівнює 1).
5. paper_indices = [i[0] for i in sim_scores]: цей рядок отримує індекси 10 найбільш подібних наукових робіт.
6. return recommend_df['abstract'].iloc[paper_indices]: останній рядок повертає тези цих 10 найбільш подібних наукових робіт.

Далі будемо отримувати рекомендації, використовуючи подібність документів за допомогою TF-IDF. В подальшому знайдемо документи, які найбільше схожі на другий абстракт у нашому наборі даних. Абстракт другого документа виглядає наступним чином:

```
1 print(recommend_df['abstract'][1])
```

```
 The formation of hot subdwarf stars (sdBs), which are core helium-burning
stars located on the extended horizontal branch, is still not understood. Many
of the known hot subdwarf stars reside in close binary systems with short
orbital periods between a few hours and a few days with either M star or white
dwarf companions. Common envelope ejection is the most probable formation
channel. Among these, eclipsing systems are of special importance because it is
possible to constrain the parameters of both components tightly by combining
spectroscopic and light curve analyses. We report the discovery of two
eclipsing binaries with a brown dwarf (< 0.07 M*) and a 0.15 M* late main
sequence star companion in close orbits around sdB stars.
```

Рисунок 3.3 - Абстракт другого документа

```
base_idx = 1
# get the first paper in the recommendation dataset
paper = recommend_df['abstract'][base_idx]
category = recommend_df['categories'][base_idx]
# Generate recommendations
print(get_recommendations(paper, cosine_sim, indices))
print('')
print('----------------------------------------------------------------------------------------------------')
similar_paper_idx = 6599
sim_paper = recommend_df['abstract'][similar_paper_idx]
sim_category = recommend_df['categories'][similar_paper_idx]
print('The article in question for recommendation has the following category assigned to it:')
print('')
print(category)
print('')
print('See the full paper abstract below:')
print('')
print(paper)
print('----------------------------------------------------------------------------------------------------')
print('----------------------------------------------------------------------------------------------------')
print('----------------------------------------------------------------------------------------------------')
print('----------------------------------------------------------------------------------------------------')
print('')
print('The article in question for recommendation has the following category assigned to it:')
print('')
print(sim_category)
print('')
print('----------------------------------------------------------------------------------------------------')
print('')
print(sim_paper)
```

Цей код використовує функцію get_recommendations, щоб знайти найбільш схожі документи на заданий абстракт. Потім він друкує абстракт і

категорію цього документа, а також абстракт і категорію одного з рекомендованих документів.

Ось детальніше:

1. base_idx = 1 - це задає індекс документа, для якого ми шукаємо рекомендації.
2. paper = recommend_df['abstract'][base_idx] і category = recommend_df['categories'][base_idx] - ці рядки отримують абстракт і категорію цього документа.
3. print(get_recommendations(paper, cosine_sim, indices)) - цей рядок викликає функцію get_recommendations, щоб отримати 10 найбільш схожих документів, і друкує їхні абстракти.
4. similar_paper_idx = 6599 - це задає індекс одного з рекомендованих документів.
5. sim_paper = recommend_df['abstract'][similar_paper_idx] і sim_category = recommend_df['categories'][similar_paper_idx] - ці рядки отримують абстракт і категорію рекомендованого документа.
6. Останні рядки коду друкують категорію і абстракт існуючого та рекомендованого документів.

Результатом (Рис. 3.4) виконання цього коду буде список абстрактів 10 найбільш схожих документів, а також детальна інформація про абстракт і категорію заданого документа та одного з рекомендованих документів.

Спробуємо отримати список з 10 абстрактами документів, які є найбільш схожими на вибраний документ з індексом 500.

```
# get the first paper in the recommendation dataset
paper = recommend_df['abstract'][500]

# Generate recommendations
print(get_recommendations(paper, cosine_sim, indices))
```

```
1781      We study moduli of holomorphic vector bundle...
49        We study (i) asymptotic behaviour of wild ha...
4121      We generalize the analysis by Moore and Witt...
467       Let G be an infinitesimal group scheme over ...
5007      Pontrjagin duality is implemented in the fra...
4380      In the present paper we describe topological...
6715      We establish a theorem computing the cohomol...
7831      We provide several results on splice-quotien...
8609      We generalize geometric prequantization of s...
4671      A fibre product construction is used to give...
Name: abstract, dtype: object
```

```
6599      A review is presented on the properties, ori...
4682      We identify SDSS011009.09+132616.1, SDSS0303...
7123      After the discovery of a substellar companio...
2659      We study the distribution of exoplanets arou...
2190      We assembled a catalogue of bright, hot subd...
9411      We present and discuss new BVI CCD photometr...
8868      Recent analyses of low-mass eclipsing binary...
8901      Observations of hot stars belonging to the y...
5031      The recently-discovered lack of close binari...
4392      Roche-lobe overflow and common envelope evol...
Name: abstract, dtype: object

--------------------------------------------------------------------------------
The article in question for recommendation has the following category assigned to it:

astro-ph.EP astro-ph.SR

See the full paper abstract below:

  The formation of hot subdwarf stars (sdBs), which are core helium-burning
stars located on the extended horizontal branch, is still not understood. Many
of the known hot subdwarf stars reside in close binary systems with short
orbital periods between a few hours and a few days with either M star or white
dwarf companions. Common envelope ejection is the most probable formation
channel. Among these, eclipsing systems are of special importance because it is
possible to constrain the parameters of both components tightly by combining
spectroscopic and light curve analyses. We report the discovery of two
eclipsing binaries with a brown dwarf (< 0.07 M*) and a 0.15 M* late main
sequence star companion in close orbits around sdB stars.

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

 --------------------------------------------------------------------------------
 --------------------------------------------------------------------------------

 The article in question for recommendation has the following category assigned to it:

 astro-ph

 --------------------------------------------------------------------------------

  A review is presented on the properties, origin and evolutionary links of hot
subluminous stars which are generally believed to be extreme Horizontal Branch
stars or closely related objects. Amongst the field stars a large fraction of
sdBs are found to reside in close binaries. The companions are predominantly
white dwarfs, or low mass main sequence stars. Systems with sufficiently
massive WD companions may qualify as SN Ia progenitors. Recently evidence has
been found that the masses of some unseen companions might exceed the
Chandrasekhar mass, hence they must be neutron stars or black holes. Even a
planet has recently been detected orbiting the pulsating sdB star V391 Peg.
Quite to the opposite, in globular clusters, only very few sdB binaries are
found indicating that the dominant sdB formation processes is different in a
dense environment. Binary population synthesis models identify three formation
channels, (i) stable Roche lobe overflow, (ii) one or two common envelope
ejection phases and (iii) the merger of two helium white dwarfs. The latter
channel may explain the properties of the He-enriched sdO stars because their
binary fraction is lower than that of the sdBs by a factor of ten or more.
Pulsating subluminous B (sdB) stars play an important role for asteroseismology
as this technique has already led to mass determinations for a handful of
stars. A unique hyper-velocity sdO star moving so fast that it is unbound to
the Galaxy has probably been ejected by the super-massive black hole in the
Galactic centre. (abridged)
```

Рисунок 3.4 - Список абстрактів 10 найбільш схожих документів

Отже, розглянуто використання TF-IDF (Term Frequency-Inverse Document Frequency) та косинусної подібності для рекомендації схожих документів на основі заданого документа. TF-IDF використовується для перетворення корпусу тексту в матрицю, яка відображає важливість кожного

терміна в кожному документі. Далі, ми використали косинусну подібність для обчислення подібності між парами документів. За допомогою функції **get_recommendations** створена генерація списку десяти найбільш схожих документів на основі абстракту заданого документа. Ми побачили, як ця функція може бути використана для отримання рекомендацій для різних документів.

3.3 Тренування моделі Doc2Vec

Виконаємо кілька задач, включаючи обчислення, природну обробку мови (NLP) та тренування моделі Doc2Vec. Спочатку, додаємо новий стовпець abstract_len до DataFrame recommend_df, який вимірює довжину (кількість слів) кожного абстракту в DataFrame. Функція describe() згодом використовується для отримання статистичного опису цих довжин. Далі завантажуємо бібліотеку NLTK (Natural Language Toolkit) і використовує її для токенізації (розбиття на окремі слова) кожного документа в наборі даних corpus. Кожному документу призначається унікальний ідентифікатор, а потім він зберігається у форматі, який може бути використаний Doc2Vec.

```
model = Doc2Vec(vector_size=100,
                alpha=0.025,
                min_alpha=0.00025,
                min_count=1,
                dm=1)

model.build_vocab(tagged_data)

for epoch in range(10):
    print('iteration {0}'.format(epoch))
    model.train(tagged_data,
                total_examples=model.corpus_count,
                epochs=10)
    # decrease the learning rate
    model.alpha -= 0.0002
    # fix th  e learning rate, no decay
    model.min_alpha = model.alpha

model.save("d2v.model")
print("Model Saved")
```

Далі відбувається створення і тренування моделі Doc2Vec. Ця модель використовується для перетворення документів на вектори, які можуть бути використані для аналізу та порівняння документів. Модель тренується на попередньо токенізованих даних з **corpus**. Модель тренується протягом 10 епох. Після кожної епохи, швидкість навчання (alpha) зменшується.

Наприкінці, модель зберігається в файлі "d2v.model" для подальшого використання. Після завершення виконання отримаємо повідомлення "Model Saved", що означає, що модель була успішно збережена.

Далі завантажуємо збережену модель Doc2Vec з файлу "d2v.model". Потім токенізуємо абстракт документа з індексом 700 з DataFrame recommend_df. Це означає, що він розбиває текст на окремі слова, переводячи його в список слів. Потім використовує метод most_similar моделі Doc2Vec для знаходження документів, що найбільше схожі на документ з тегом '1'. Цей метод порівнює вектори документів і повертає найбільш схожі на основі косинусної подібності.

Результати (Рис. 3.5) (найбільш схожі документи) друкуються на екрані. Кожен результат представляється у формі (тег, подібність), де тег - це

унікальний ідентифікатор документа, а подібність - це міра подібності (від 0 до 1) до вхідного документа. Отже, у вашому випадку:

- Документ з тегом '6071' є найбільш схожим на вхідний документ, із подібністю 0.5418.
- Документ з тегом '2410' є наступним за ступенем подібності, із подібністю 0.5298.
- І так далі, знижуючи ступінь подібності.

```
similar : [('6071', 0.5418484210968018), ('2410', 0.5298833847045898), ('4682', 0.51924222707748
41), ('4777', 0.5181699991226196), ('2695', 0.5085140466690063), ('6599', 0.5077699422836304),
('9083', 0.49743303656578064), ('6219', 0.49511611461639404), ('5205', 0.49097663164138794), ('3
020', 0.4882623851299286)]
```

Рисунок 3.5 - Найбільш схожі документи до документ з тегом '1'

```
similar_paper_idx = 6071
sim_paper = recommend_df['abstract'][similar_paper_idx]
sim_category = recommend_df['categories'][similar_paper_idx]
print('The article in question for recommendation has the following category assigned to it:')
print(category)
print('')
print('See the full paper abstract below:')
print('')
print(recommend_df['abstract'][1])
print('-------------------------------------------------------------------------------------------')
print('-------------------------------------------------------------------------------------------')
print('-------------------------------------------------------------------------------------------')
print('-------------------------------------------------------------------------------------------')
print('')
print('The article in question for recommendation has the following category assigned to it:')
print(sim_category)
print('')
print('-------------------------------------------------------------------------------------------')
print('-------------------------------------------------------------------------------------------')
print('-------------------------------------------------------------------------------------------')
print('-------------------------------------------------------------------------------------------')
print('')
print(sim_paper)
```

Цей код виводить інформацію про два наукові статті: одну, що була використана як вхідний документ для порівняння (з індексом 1), та іншу, що була визначена як найбільш схожа на вхідний документ (з індексом 6071).

Він виводить наступну інформацію:

1. Категорію вхідного документа (того, що було використано для порівняння). Змінна category, яка використовується тут, не була визначена в наведеному коді, але ми можемо припустити, що вона містить категорію вхідного документа.
2. Повний абстракт вхідного документа.
3. Категорію найбільш схожого документа, знайденого за допомогою моделі Doc2Vec.
4. Повний абстракт найбільш схожого документа (Рис. 3.6).

Реалізація модуля підбору тематики наукових статей на основі NLP використовує Doc2Vec для аналізу текстів. Алгоритм токенізує, тренує модель та знаходить схожі статті. Результати можуть бути використані для рекомендацій, класифікації або кластеризації.

```
The article in question for recommendation has the following category assigned to it:
astro-ph.EP astro-ph.SR

See the full paper abstract below:

  The formation of hot subdwarf stars (sdBs), which are core helium-burning
stars located on the extended horizontal branch, is still not understood. Many
of the known hot subdwarf stars reside in close binary systems with short
orbital periods between a few hours and a few days with either M star or white
dwarf companions. Common envelope ejection is the most probable formation
channel. Among these, eclipsing systems are of special importance because it is
possible to constrain the parameters of both components tightly by combining
spectroscopic and light curve analyses. We report the discovery of two
eclipsing binaries with a brown dwarf (< 0.07 M*) and a 0.15 M* late main
sequence star companion in close orbits around sdB stars.

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

The article in question for recommendation has the following category assigned to it:
astro-ph

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

  We present the discovery and CCD observations of the first eclipsing binary
with a Type II Cepheid component in our Galaxy. The pulsation and orbital
periods are found to be 4.1523 and 51.38 days, respectively, i.e. this variable
is a system with the shortest orbital period among known Cepheid binaries.
Pulsations dominate the brightness variations. The eclipses are assumed to be
partial. The EB-subtype eclipsing light curve permits to believe that the
binary's components are non-spherical.
```

Рисунок 3.6 - Повний абстракт найбільш схожого документа за допомогою моделі Doc2Vec

3.4 Кластеризація документів

Для кластеризації документів, на першому етапі використаємо метод головних компонент (PCA) з бібліотеки scikit-learn для зменшення розмірності даних.

```
pca = PCA(n_components=0.95)
df_reduced = pca.fit_transform(tfidf_matrix.toarray())
df_reduced
```

У вашому коді вказано n_components=0.95, що означає, що PCA вибере мінімальну кількість компонентів так, щоб зберегти 95% дисперсії даних.

Метод fit_transform() обчислює головні компоненти на основі даних (tfidf_matrix.toarray()), а потім використовує ці компоненти для трансформації вихідних даних до нового простору з меншою розмірністю.

Результатом є матриця df_reduced (Рис.3.7), де кожен рядок відповідає вихідному рядку в tfidf_matrix, але кількість стовпців зменшилась до кількості головних компонент, які змогли зберегти 95% варіативності вихідних даних. Це може бути корисно для візуалізації даних, або як попередній крок до використання алгоритмів машинного навчання, які можуть краще працювати з даними меншої розмірності. Кожне число в масиві представляє вагу відповідної головної компоненти для конкретного вхідного вектора. Ці ваги показують, наскільки сильно вхідні дані відображаються вздовж кожної з головних компонент.

```
array([[ 0.04681021, -0.06006718, -0.07293093, ...,  0.01363775,
        -0.00604095, -0.00171744],
       [-0.0015821 ,  0.15761229, -0.05595536, ...,  0.00681738,
         0.00016558,  0.00796251],
       [ 0.00212261, -0.03539648, -0.03991771, ..., -0.00396655,
        -0.00803769, -0.00188201],
       ...,
       [ 0.08355614, -0.07911495,  0.02814823, ..., -0.00230786,
        -0.00522723, -0.00175423],
       [-0.05269326, -0.0259181 , -0.02859099, ..., -0.00038912,
        -0.00111854,  0.00115527],
       [-0.14962038, -0.02074884, -0.01983241, ..., -0.00295365,
         0.00360968, -0.00742522]])
```

Рисунок 3.7 - Матриця df_reduced для PCA

Далі проводимо кластеризацію даних за допомогою алгоритму K-середніх (KMeans) і досліджуємо, як змінюється внутрішня сума квадратів помилок (SSE, англ. Sum of Squared Errors) в залежності від кількості кластерів.

```
kmeans_kwargs = {
"init": "random",
# "n_init": 10,
"random_state": 1,
}

#create list to hold SSE values for each k
sse = []
for k in range(1, 500, 50):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(tfidf_matrix)
    sse.append(kmeans.inertia_)

#visualize results
values = range(1, 500, 50)
plt.plot(values, sse)
plt.xticks(values)
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title('Clustering SSE by # of Clusters (Elbow-Method)')
plt.show()
```

У цьому коді проводиться кластеризація для кількості кластерів від 1 до 500 з кроком 50. Для кожного значення k обчислюється SSE (доступний як kmeans.inertia_ після підгонки моделі KMeans) і зберігається в списку sse.

Нарешті, код візуалізує (Рис.3.8.) залежність SSE від кількості кластерів, створюючи графік. Це робиться для визначення оптимальної кількості кластерів за методом "ліктя". Ідея цього методу полягає в тому, що на графіку SSE від кількості кластерів буде видно "лікоть" - точку, після якої збільшення кількості кластерів призводить до незначного зменшення SSE. Ця точка і є рекомендованою кількістю кластерів.

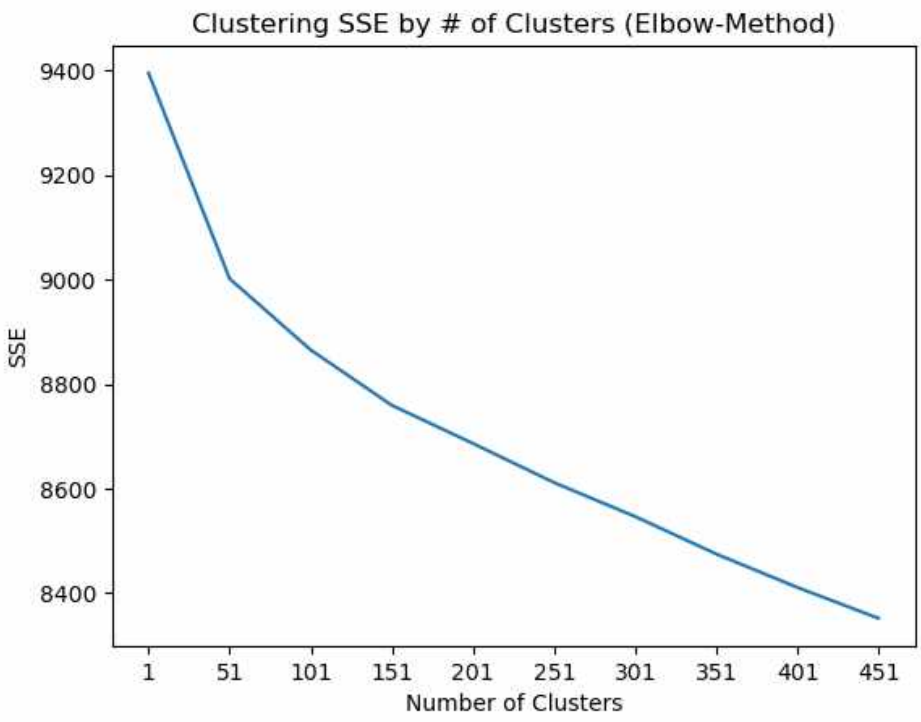


Рисунок 3.8 - Кластеризація SSE за кількістю кластерів (Метод ліктя)

```
kmeans_kwargs = {
"init": "random",
# "n_init": 10,
"random_state": 1,
}

#create list to hold SSE values for each k
sse = []
for k in range(1, 100, 20):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(df_reduced)
    sse.append(kmeans.inertia_)

#visualize results
values = range(1, 100, 20)
plt.plot(values, sse)
plt.xticks(values)
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title('Clustering SSE by # of Clusters (Elbow-Method)')
plt.show()
```

Цей код дуже схожий на попередній, але з однією ключовою відмінністю: він використовує дані **df_reduced**, замість **tfidf_matrix** для кластеризації KMeans.

df_reduced - це дані, які були попередньо трансформовані за допомогою методу головних компонент (PCA), тому вони мають меншу розмірність, ніж оригінальні дані **tfidf_matrix**. Використання **df_reduced** замість **tfidf_matrix** може мати декілька переваг. По-перше, зменшення розмірності може прискорити обчислення, особливо при великій кількості кластерів. По-друге, використання PCA може допомогти зосередитися на найважливіших аспектах даних і відкинути менш важливі, що може привести до кращої кластеризації.

Також зазначимо, що діапазон значень **k** (кількість кластерів) та крок змінилися. У цьому коді **k** змінюється від 1 до 100 з кроком 20, тоді як у попередньому коді **k** змінювався від 1 до 500 з кроком 50. Це може вплинути на знайдену оптимальну кількість кластерів.

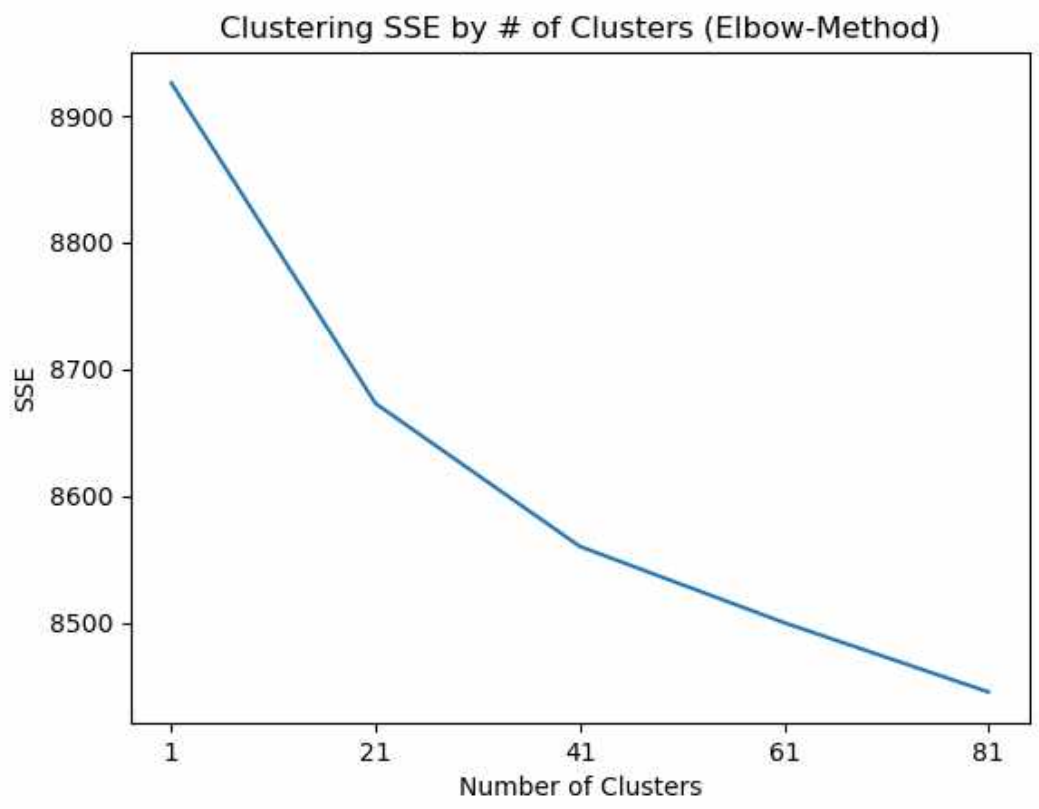


Рисунок 3.9 - Кластеризація SSE за кількістю кластерів (Метод ліктя) на основі PCA

Отже, створимо модель кластеризації K-середніх (KMeans) з 20 кластерами (це значення k було визначено на основі попереднього аналізу "методу ліктя") і застосовує її до даних df_reduced.

```
k = 20 # optimal k found in elbow plot
kmeans = KMeans(n_clusters=k, random_state=42)
y_pred = kmeans.fit_predict(df_reduced)
```

kmeans.fit_predict(df_reduced) робить дві речі: во-перше, вона "навчає" модель на вхідних даних (тобто визначає центроїди кластерів), а потім вона використовує ці центроїди, щоб визначити, до якого кластера належить кожна точка вхідних даних.

Результатом є масив y_pred, де кожен елемент відповідає точці вхідних даних і є номером кластера, до якого ця точка була віднесена. Наприклад, якщо y_pred[0] == 3, це означає, що перша точка вхідних даних була віднесена до кластера номер 3.

Використання random_state=42 гарантує, що результати будуть відтворюваними: при кожному запуску цього коду ініціалізація центроїдів кластерів буде однаковою, що приведе до однакового результату кластеризації.

Створимо новий DataFrame df, що об'єднує три компоненти:

1. pd.Series(corpus): це створює pandas Series з корпусу текстових даних. Корпус - це колекція текстових документів, і в даному контексті, це могли б бути, наприклад, статті, відгуки користувачів, твіти тощо. Цей рядок буде першим стовпцем нового DataFrame.
2. pd.Series(y_pred, name='cluster'): це створює pandas Series з масиву y_pred, який містить прогнози кластерів для кожного документа в корпусі. Цей рядок буде другим стовпцем нового DataFrame і буде названий 'cluster'.

3. recommend_df['title'].head(10000): це вибирає перші 10000 рядків з стовпця 'title' DataFrame recommend_df. Це можуть бути, наприклад, назви документів у корпусі. Цей рядок буде третім стовпцем нового DataFrame.

Функція pd.concat(..., axis=1) об'єднує ці три рядки вздовж осі стовпців (тобто, вони стають стовпцями нового DataFrame, а не рядками).

Таким чином, новий DataFrame df (Таблиця 3.1) містить три стовпці: текст кожного документа, номер кластера, до якого він був віднесений, та його назву (якщо вона доступна). Це може бути корисним для подальшого аналізу документів та кластерів.

Таблиця 3.1. Розподіл кластерів

	abstract	cluster	title
0	In this paper we investigate the dispersive ...	15	Strichartz Estimates for Water Waves
1	The formation of hot subdwarf stars (sdBs), ...	2	Analysis of Two Eclipsing Hot Subdwarf Binarie...
2	The k-nearest-neighbour procedure is a well-...	0	A Bayesian reassessment of nearest-neighbour c...
3	The proposal is to use clusters, graphs and ...	18	Clusters, Graphs, and Networks for Analysing l...
4	The heat channel is defined by a linear time...	15	On the Capacity of the Heat Channel, Waterfill...
...	...	...	...
9995	We present the SINS survey with SINFONI of h...	17	The SINS survey: SINFONI Integral Field Spectr...
9996	A BCS (Bardeen-Cooper-Schrieffer) supercondu...	19	Multi-condensate states in BCS superconductors
9997	The order of the phase transition in finite-...	9	Universality and Scaling at the chiral transit...
9998	A summary introduction of the Weil-Petersson...	19	The Weil-Petersson metric geometry
9999	A central arrangement \A of hyperplanes in...	1	Totally free arrangements of hyperplanes

```
tsne = TSNE(verbose=1, perplexity=100, random_state=42)
X_embedded = tsne.fit_transform(df_reduced)
```

Цей код використовує t-SNE (t-Distributed Stochastic Neighbor Embedding), що є технікою зменшення розмірності особливо підходящою для візуалізації високовимірних даних. t-SNE перетворює високовимірні дані таким чином, що подібні об'єкти моделюються як близькі точки в двох- або тривимірному просторі, а різні об'єкти - як далекі точки.

У цьому коді tsne.fit_transform(df_reduced) навчає модель t-SNE на даних df_reduced і застосовує перетворення до цих даних. Результатом є двовимірне представлення даних df_reduced, яке зберігається в X_embedded.

Параметр perplexity впливає на баланс між збереженням маленьких та великих відстаней, а random_state забезпечує відтворюваність результатів.

Отже, створимо двовимірну діаграму розсіювання на основі результатів t-SNE та міток KMeans (Рис.3.10).

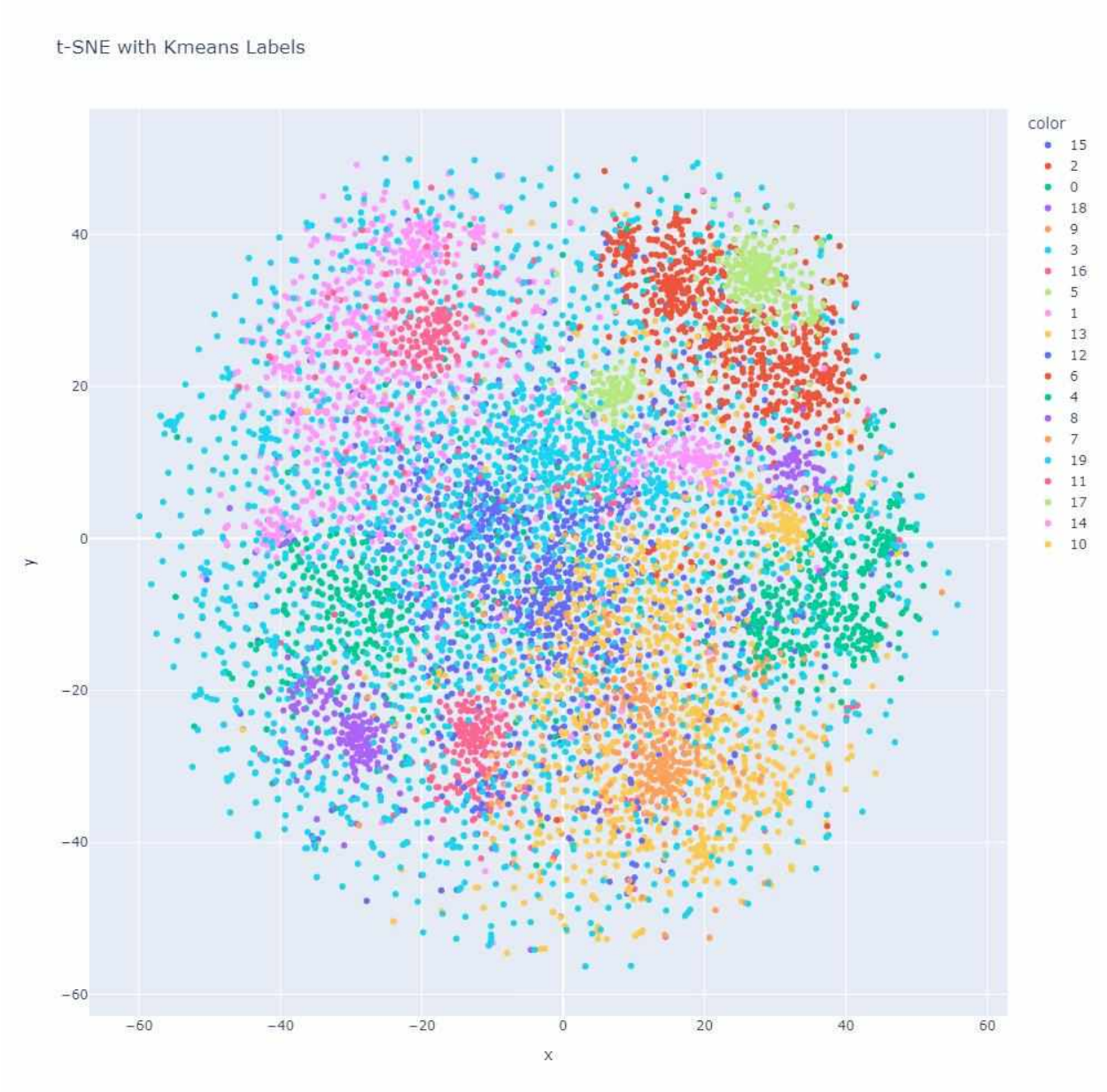


Рисунок 3.10 - Діаграма розсіювання на основі результатів t-SNE та міток KMeans

Вище представлений інтерактивний графік, який відображає низьковимірне представлення наукових статей у наборі даних. Коли ви наводите курсор на точки даних, відображається назва статті. Ви можете двічі клацнути на точках у легенді, щоб відобразити лише точки з одного кластера, а потім вибрати додаткові кластери звідти для безпосереднього порівняння.

Такий візуальний засіб дозволяє краще зрозуміти, як статті групуються в кластери на основі їхнього вмісту. Ви можете побачити, які кластери виокремлюються ясно, а які мають більше перекриття з іншими. Це також дає можливість глибше зануритися в окремі статті, переглядаючи їх назви при наведенні курсора.

ВИСНОВКИ

Модуль підбору тематики наукових статей на основі NLP використовує різноманітні методи та техніки для аналізу тексту з метою визначення тематичної спрямованості статей. Це сприяє збільшенню ефективності та точності підбору статей для конкретних задач та досліджень. Алгоритм обробки та розподілу набору даних для подальшого текстового аналізу є важливою складовою модуля, оскільки забезпечує оптимальну підготовку даних перед застосуванням методів обробки природної мови (NLP). Цей алгоритм дозволяє підготувати дані перед застосуванням NLP-методів для виявлення тематики наукових статей, забезпечуючи належну якість даних та структурованість текстових документів.

Алгоритм, розроблений для модуля, використовує технологію Doc2Vec для виявлення схожості між науковими статтями. Його тренування на корпусі текстів дозволяє виявляти семантичні зв'язки між документами, що поліпшує здатність системи підбору тематики до більш точного та ефективного рекомендування, класифікації та кластеризації статей.

Алгоритм кластеризації документів, який базується на NLP та машинному навчанні, використовує TF-IDF, PCA та KMeans для групування статей за темами. Потім візуалізація цих груп в двовимірному просторі за допомогою t-SNE допомагає зрозуміти залежності та взаємозв'язки між ними.

У даній роботі було розглянуто використання TF-IDF та косинусної подібності для рекомендації схожих документів на основі заданого документа. TF-IDF дозволяє визначити важливість термінів в кожному документі, а косинусна подібність використовується для обчислення ступеня схожості між документами. За допомогою функції get_recommendations було реалізовано генерацію списку найбільш схожих документів на основі абстракту заданого документа.

Також була розглянута реалізація модуля підбору тематики наукових статей на основі NLP з використанням алгоритму Doc2Vec. Цей алгоритм забезпечує аналіз текстових документів, включаючи тренування моделі та пошук схожих статей. Отримані результати можуть бути використані для рекомендацій, класифікації або кластеризації статей.

Також був представлений інтерактивний графік, що відображає низьковимірне представлення наукових статей у наборі даних. Цей графік надає візуальне уявлення про групування статей у кластери на основі їх вмісту. Це дозволяє краще зрозуміти, як статті пов'язані між собою та які кластери виокремлюються чітко або мають перекриття з іншими. Такий підхід дозволяє глибше вивчати окремі статті та знаходити залежності між ними.

Усі ці підходи та методи виявляються ефективними для підбору тематики наукових статей на основі NLP. Вони допомагають збільшити точність та ефективність підбору статей для конкретних задач або досліджень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gaikwad, S.V., Chaugule, A., Patil, P.: "Text mining methods and techniques." Int. J. Comput. Appl. 85(17), 2014.
2. Salloum, S.A., Al-Emran, M., Monem, A.A., Shaalan, K.: "A Survey of text mining in social media: Facebook and Twitter perspectives." Adv. Sci. Technol. Eng. Syst. J., 2017.
3. Navathe, S.B., Ramez, E.: "Data warehousing and data mining." Fundam. Database Syst., pp. 841-872, 2000.
4. Gupta, V., Lehal, G.S.: "A survey of text mining techniques and applications." J. Emerg. Technol. Web Intell. 1(1), pp. 60-76, 2009.
5. Gupta, S., Kaiser, G.E., Grimm, P., Chiang, M.F., Starren, J.: "Automating content extraction of HTML documents." World Wide Web 8(2), pp. 179-224, 2005.
6. Hassani, H., Huang, X., Silva, E.S., Ghodsi, M.: "A review of data mining applications in crime." Statistical Anal. Data Min.: ASA Data Sci. J. 9(3), pp. 139-154, 2016.
7. Feldman, R., Dagan, I.: "Knowledge discovery in textual databases (KDT)." KDD 95, pp. 112-117, 1995.
8. Tan, A.H.: "Text mining: The state of the art and the challenges." In: Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases, vol. 8, pp. 65-70, 1999.
9. Hearst, M.A.: "Untangling text data mining." In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics, pp. 3-10. Association for Computational Linguistics, 1999.
10. Rajman, M., Besançon, R.: "Text mining: natural language techniques and text mining applications." In: Data Mining and Reverse Engineering, pp. 50-64. Springer, US, 1998.

11. Mahgoub, H., Rösner, D., Ismail, N., Torkey, F.: "A text mining technique using association rules extraction." Int. J. Computat. Intell. 4(1), pp. 21-28, 2008.
12. Akilan, A.: "Text mining: challenges and future directions." In: 2015 2nd International Conference on Electronics and Communication Systems (ICECS), pp. 1679-1684. IEEE, 2015.
13. Sukanya, M., Biruntha, S.: "Techniques on text mining." In: 2012 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), pp. 269-271. IEEE, 2012.
14. Salloum, S.A., Al-Emran, M., Shaalan, K.: "A Survey of lexical functional grammar in the Arabic context." Int. J. Com. Net. Tech. 4(3), 2016.
15. Al-Emran, M., Shaalan, K.: A survey of intelligent language tutoring systems. In: 2014 International Conference on Advances in Computing, Communications and Informatics ICACCI, pp. 393–399. IEEE (2014a)
16. Al-Emran, M., Zaza, S., Shaalan, K.: Parsing modern standard Arabic using Treebank resources. In: 2015 International Conference on Information and Communication Technology Research (ICTRC), pp. 80–83. IEEE (2015)
17. Pazienza, M.T. (Ed.): Information extraction: Towards scalable, adaptable systems. Springer (2003)
18. Cowie, J., Lehnert, W.: Information extraction. Commun. ACM 39(1), 80–91 (1996)
19. Velasco-Elizondo, P., Marín-Piña, R., Vazquez-Reyes, S., Mora-Soto, A., Mejia, J.: Knowledge representation and information extraction for analyzing architectural patterns. Sci. Comput. Program. 121, 176–189 (2016)
20. Hsu, J.Y.J., Yih, W.T.: Template-based information mining from HTML documents. In: AAAI/IAAI, pp. 256–262 (1997)
21. Mooney, R.J., Nahm, U.Y.: Text mining with information extraction, multilingualism, and electronic language management. In: Proceedings 4th International MIDP Colloquium, pp. 141–160 (2003)

22. Clifton, C., Cooley, R., Rennie, J.: TopCat: data mining for topic identification in a text corpus. IEEE Trans. Knowl. Data Eng. 16(8), 949–964 (2004)
23. Sirsat, S.R., Chavan, D.V., Deshpande, D.S.P.: Mining knowledge from text repositories using information extraction: A review. Sadhana 39(1), 53–62 (2014)
24. Madani, F.: Technology Mining bibliometrics analysis: applying network analysis and cluster analysis. Scientometrics 105(1), 323–335 (2015)
25. Huang, A.: Similarity measures for text document clustering. In: Proceedings of the sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand, pp. 49–56 (2008)
26. Clifton, C., Cooley, R.: TopCat: Data mining for topic identification in a text corpus. In: European Conference on Principles of Data Mining and Knowledge Discovery, pp. 174–183. Springer, Heidelberg (1999)
27. Han, E.H., Karypis, G., Kumar, V., Mobasher, B.: Clustering based on association rule hypergraphs. In: DMKD (1997)
28. Irfan, R., King, C.K., Grages, D., Ewen, S., Khan, S.U., Madani, S.A., & Tziritas, N.: A survey on text mining in social networks. Knowl. Eng. Rev. 30(2), 157–170 (2015)
29. Goh, D.H., Ang, R.P.: An introduction to association rule mining: An application in counseling and help-seeking behavior of adolescents. Behav. Res. Methods 39(2), 259–266 (2007)
30. Wong, P.C., Whitney, P., Thomas, J.: Visualizing association rules for text mining. In: 1999 IEEE Symposium on Information Visualization, 1999.
30. Li, G., Liu, H., Hu, X., Yu, P.S.: Topic mining with temporal constraints. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 262–271. ACM (2004)
31. Song, J., Li, Y., Zhang, J., Li, Y.: A novel approach to document clustering based on frequent itemsets. In: Proceedings of the 7th Pacific-Asia

Conference on Knowledge Discovery and Data Mining (PAKDD 2003), pp. 624–635. Springer (2003)

32. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD International Conference on Management of Data, pp. 1–12. ACM (2000)
33. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: 20th International Conference on Very Large Data Bases, VLDB 1994, Santiago de Chile, Chile, September 12-15, 1994, Proceedings, pp. 487–499. Morgan Kaufmann (1994)
34. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manag. 24(5), 513–523 (1988)
35. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.A.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. 41(6), 391–407 (1990)
36. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
37. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
38. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751. Association for Computational Linguistics (2014)
39. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196 (2014)
40. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)

41. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2nd edn. Prentice Hall, Upper Saddle River (2008)
42. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (1999)
43. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media, Sebastopol (2009)
45. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55–60. Association for Computational Linguistics (2014)
46. Jurafsky, D., Martin, J.H.: An Introduction to Natural Language Processing. Draft of an upcoming textbook, Stanford University (2021). [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/
47. Socher, R., Perelygin, A., Wu, J.Y., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1631–1642. Association for Computational Linguistics (2013)
48. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. 12, 2493–2537 (2011)
49. Goldberg, Y.: A primer on neural network models for natural language processing. J. Artif. Intell. Res. 57, 345–420 (2016)
50. Manning, C.D.: Deep learning for natural language processing. In: Handbook of Natural Language Processing, pp. 69–88. Chapman and Hall/CRC (2010)

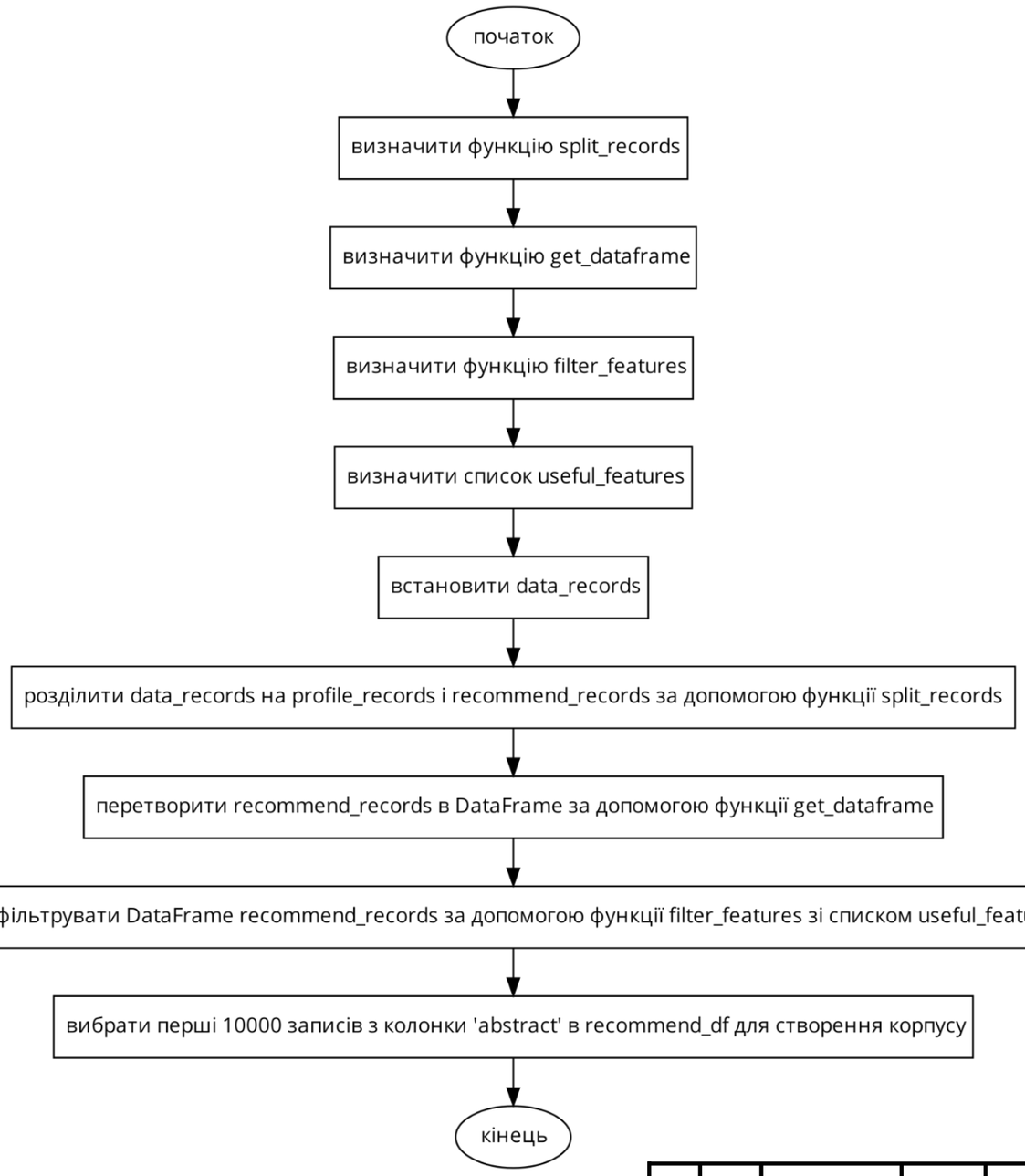
51. Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing. IEEE Comput. Intell. Mag. 13(3), 55–75 (2018)
52. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
53. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 4171–4186. Association for Computational Linguistics (2019)
54. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
55. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog 1(8), 9 (2019)
56. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Liu, Y., Jiang, X., Gouws, S., Kolesnikov, S., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683 (2019)
57. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
58. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)
59. Vaswani, A., Zhao, Y., Fossum, V., Chaudhary, N., Blackwood, G., Cai, S., Cassol, F., Choksi, K., Deng, L., Deng, Z., et al.: Pegasus: Pre-training with

extracted gap-sentences for abstractive summarization. arXiv preprint arXiv:1912.08777 (2019)

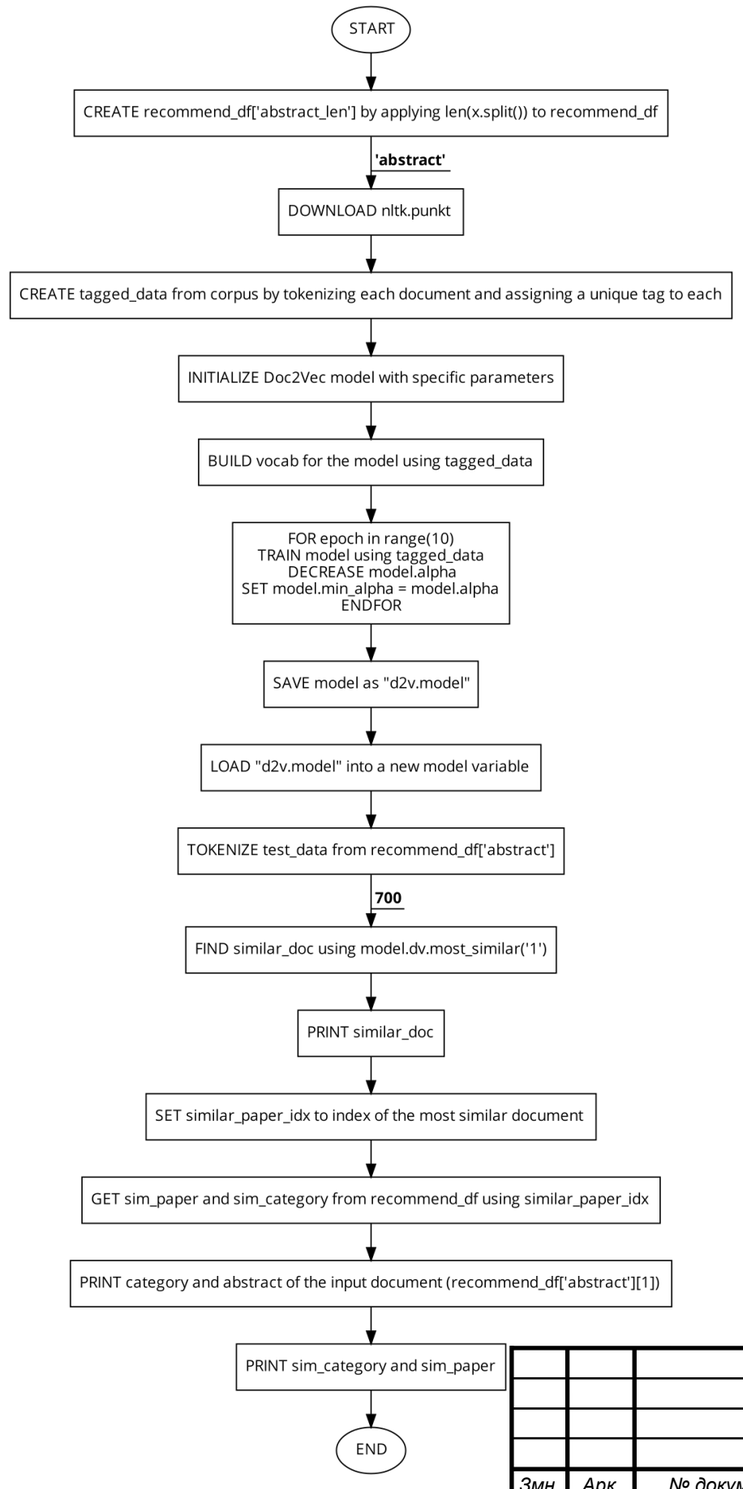
60. Ruder, S., Howard, J., Recht, B.: Universal language model fine-tuning for text classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, pp. 328–339 (2018)
61. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Pre-trained language models for diverse tasks. arXiv preprint arXiv:1904.07743 (2019)
62. Liu, Y., Liu, M., Gardner, M., Zettlemoyer, L., Eisenstein, J.: Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. arXiv preprint arXiv:1911.11931 (2019)
63. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461 (2019)
64. Liu, Y., Kitaev, N., Chen, K., Levy, O., Goyal, N., Ostendorf, M., Jurafsky, D.: RoFormer: Enhanced transformer with rotary position embedding. arXiv preprint arXiv:2104.09864 (2021)
65. Brown, T.B., Biderman, S., Wei, W., Weston, J., Senior, A., Bakhtin, A., Painter, P., Kaggle, P., Fergus, R.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
66. V. Krylov et al., "Multiple Regression Method for Analyzing the Tourist Demand Considering the Influence Factors," 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Metz, France, 2019, pp. 974-979, doi: 10.1109/IDAACS.2019.8924461.
67. Lipyanina, H., Sachenko, S., Lendyuk, T., Brych, V., Yatskiv, V., & Osolinskiy, O. (2021). Method of detecting a fictitious company on the

machine learning base. In Advances in Computer Science for Engineering and Education IV (pp. 138-146). Cham: Springer International Publishing.

68. Gramyak, R., Lipyanina-Goncharenko, H., Sachenko, A., Lendyuk, T., & Zahorodnia, D. (2021). Intelligent Method of a Competitive Product Choosing based on the Emotional Feedbacks Coloring. In IntelITSIS (pp. 246-257).
69. *arXiv Dataset*. (б. д.). Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com/datasets/Cornell-University/arxiv
70. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С. Методичні рекомендації до виконання кваліфікаційної роботи з освітньопрофесійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. – Тернопіль: ЗУНУ, 2021. – 56 с.



					ДП.КН. 8091523.001 А1			
					Алгоритм обробки та розподілі набору даних для подальшого текстового аналізу	Літера	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Грицай Р.І.				Аркуш 1	Аркушів 1	
Перевір.		Лендюк Т.В						
Консультант						ЗУНУ.ФКІТ.КН-41		
Т. Контр.								
Н. Контр.		Лендюк Т.В						
Затверд.		Комар М.П.						



					ДП.КН. 8091523.001 А1	
Змн.	Арк.	№ докум.	Підпис	Дата	Алгоритм тренування моделі Doc2Vec для підбору тематики наукових статей	Літера / Маса / Масштаб
Розроб.		Грицай Р.І.				
Перевір.		Лендюк Т.В				Аркуш 1 / Аркушів 1
Консультант						
Т. Контр.						ЗУНУ.ФКІТ.КН-41
Н. Контр.		Лендюк Т.В				
Затверд.		Комар М.П.				