Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Факультет інформаційних технологій
(факультет)

Кафедра системного аналізу та управління
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

Студента Ємця Миколи Олександровича
академічної групи 124–20–1
спеціальності 124 Системний аналіз
на тему: «Генерація ключових слів відео за текстовим описом на основі штучного інтелекту»

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**Інституційною**	
кваліфікаційної роботи	*асист. Хабарлак К.С.*			
розділів:				
Інформаційно–аналітичний	*асист. Хабарлак К.С.*			
Спеціальний розділ	*асист. Хабарлак К.С.*			
Рецензент				
Нормоконтролер	*к.ф.–м.н., доц. Хом'як Т.В.*			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри
Системного аналізу та управління
(повна назва)

_____ *к.т.н., доц. Желдак Т.А.*
(підпис) (прізвище,ініціали)

«____» _____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня *бакалавра*

студенту *Ємцю М.О.* **академічної групи** *124–20–1*
спеціальності: *124 Системний аналіз*
на тему «*Генерація ключових слів відео за текстовим описом на основі штучного інтелекту»*
затверджену наказом ректора НТУ «Дніпровська політехніка»
від *23.05.2024 р. №469-с*

Розділ	Зміст	Терміни виконання
1. Інформаційно-аналітичний розділ	Проаналізувати структуру об'єкта дослідження. Визначити предметну область дослідження та проблему, що розв'язується. Обґрунтувати методи виконання поставлених завдань.	11.09.2023-28.01.2024
2.Спеціальний розділ	Розв'язати поставлені задачі: 1) первинний аналіз метаданих відео з платформи YouTube; 2) порівняльний аналіз якості методів створення векторного представлення слів, багатокласової класифікації на зазначеному наборі даних; 3) програма реалізація застосунку генерації ключових слів відео за коротким текстовим описом для підвищення охоплення та популярності відео освітньої, наукової тематики.	29.01.2024-13.06.2024

Завдання видано _____ *Хабарлак К.С.*
(підпис) (прізвище, ініціали)

Дата видачі: *04.09.2023*
Дата подання до екзаменаційної комісії: _____
Прийнято до виконання _____ Ємець М.О.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 76 с., 18 рис., 14 табл., 6 додатків, 14 джерел.

Об'єктом дослідження в роботі є процес розробки технології, що допомагає популяризувати відео освітньої тематики на платформі YouTube.

Предметом дослідження є алгоритм вирішення задач класифікації текстової інформації з використанням штучного інтелекту.

Метою даної кваліфікаційної роботи є підвищення популярності відео освітньої тематики шляхом розробки програмного додатку, який автоматично генеруватиме ключові слова на основі текстового опису відео що сприятиме його охопленню більшою кількості зацікавлених глядачів.

Методи дослідження: порівняльний метод, для порівняння ефективності використаних інструментів. Метод класифікації на основі машинного навчання для генерації ключових слів відео, на основі його опису.

В інформаційно–аналітичному розділі наведено сутність задачі класифікації текстової інформації, актуальні приклади вирішення. Визначені необхідні інструменти для вирішення поставленої задачі.

У спеціальному розділі здійснена програмна реалізація генерації ключових слів відео за його описом. Проведено аналіз отриманих результатів, визначено оптимальне рішення.

Практична цінність отриманих результатів полягає в тому, що запропонований додаток дозволить популяризувати відео освітньої тематики, шляхом збільшення популярності за рахунок оптимально підібраних ключових слів.

Ключові слова: NLP, КЛАСИФІКАЦІЯ ТЕКСТУ, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ

ABSTRACT

Explanatory note: 76 p., 18 fig., 14 tables, 6 appendices, 14 sources.

The object of research in the work includes the process of developing a technology that helps popularize scientific videos on the YouTube platform.

The subject of research is an algorithm for solving text information classification problems.

The purpose of this qualification work is to increase the popularity of educational videos by developing a software application that will automatically generate keywords based on the text description of the video, which will help it reach more interested viewers.

Research methods: a comparative method to compare the effectiveness of text representation methods and machine learning algorithms. A machine learning based classification method for keyword generation.

The *information and analytical section* contain the essence of the task of classifying textual information and actual examples of its solution. The necessary tools for solving the task are identified.

In the *special section* software implementation of video keyword generation based on its description. The results were analyzed and the optimal solution was determined.

The practical value of the results obtained is that the proposed application will allow you to popularize videos on scientific topics by increasing their popularity through optimally selected keywords.

Keywords: NLP, TEXT CLASSIFICATION, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING

ЗМІСТ

ВСТУП

У сучасному світі, де обсяг інформації невпинно зростає, пошук релевантного контенту стає дедалі складнішим завданням. Це особливо актуально для наукових відео на платформі YouTube, де конкуренція за увагу глядачів надзвичайно висока. Одним із важливих аспектів популяризації таких відео є правильне використання ключових слів, що дозволяє відео бути знайденим та переглянутим цільовою аудиторією.

Актуальність теми дослідження обумовлена необхідністю автоматизації процесу генерації ключових слів для відео наукової тематики. Автоматизація дозволить значно скоротити час та зусилля, які витрачаються на визначення ключових слів людиною, а також допоможе уникнути людського фактору, за рахунок якого ключові слова можуть бути визначені хибно. Залучення штучного інтелекту до вирішення цієї задачі надає можливість користувачу заощадити час при підборі ключових слів, а також обрати ті, які найточніше б описували його відео та надали змогу залучити більше аудиторії шляхом спрощення пошуку цього відео.

Метою цієї дипломної роботи є підвищення популярності відео освітньої тематики шляхом розробки програмного додатку, який автоматично генеруватиме ключові слова на основі текстового опису відео. Це сприятиме кращому позиціонуванню відео на платформі, збільшенню його видимості для цільової аудиторії та охопленню більшої кількості зацікавлених глядачів.

Результати даної роботи мають практичну користь, оскільки вони можуть бути використані для автоматизації процесу оптимізації відео контенту на YouTube, що сприятиме більш ефективній популяризації наукових знань та залученню більшої аудиторії.

Тези доповіді щодо результатів досліджень було подано до І (VII) міжнародна науково-практичної конференції здобувачів вищої освіти і молодих учених «Інформаційні технології: теорія і практика» (20 – 22 березня 2024 р). Збірник тез. Дніпро. НТУ «ДП». – 168-170 с.

1. ОСОБЛИВОСТІ ЗАДАЧІ ГЕНЕРАЦІЇ КЛЮЧОВИХ СЛІВ

1.1 Аналіз рішень задач класифікації тексту

Задача генерації ключових слів за описом відео стосується загального напрямку штучного інтелекту та математичної лінгвістики – обробки тексту природньої мови, або NLP (Natural Language Processing). NLP – це область науки, яка поєднує в собі два напрямки: гуманітарну лінгвістику та інноваційні технології штучного інтелекту. Основною метою є створення умов для розуміння машиною сенсу людської мови.

На цьому етапі виникає багато перешкод, таких як:

- Наявність осмисленості мови, яка відображається в передачі не тільки звуків та символів, але й інформації, яку необхідно інтерпретувати.
- Велика кількість параметрів для обробки, для досягнення розуміння машиною сенсу текстового повідомлення.
- Наявність синтаксичних, граматичних та лексичних нюансів, які значно ускладнюють сприйняття машиною тексту. Наприклад слова омографи.
- Наявність жаргонізмів, скорочень слів, неологізмів, професійної та фольклорної лексики. Словарний запас постійно поновлюється, що зумовлює необхідність неперервного навчання машинних алгоритмів.

Можливість машини зрозуміти природню мову потребує значної обізнаності системи в навколишньому світі для взаємодії з інформацією та розуміння повного її контексту.

На сьогоднішній день, NLP включає вирішення багатьох задач, таких як: розпізнавання мови, аналіз тексту (отримання необхідної інформації, аналіз тональності тексту), генерація тексту на основі запитів, синтез мови, машинний переклад, автоматичне анотування, категоризація.

Існує два методи класифікації тексту:

1) Метод класифікації на основі правил. В цьому випадку розробляються спеціально розроблені людиною лінгвістичні правила, які використовує

система для значення належності фрагменту тексту до певного класу. Система шукає підказки в виді семантично релевантних текстових елементів. У кожного правила наявний свій шаблон, якому має відповідати текст, аби потрапити у певну категорію. Цей метод використовується для вирішення простих задач, таких як визначення погоди за описом, або визначення типу товару в магазині за його характеристиками [2].

Перевагами методу є передбачуваність вхідної та вихідної інформації та можливість їх інтерпретації, а також можливість покращення моделі шляхом втручання розробника.

Недоліками методу є ненадійність моделі, яка може тільки притримуватися заданого шаблону. Саме тому використання цього методу для вирішення поставленої задачі не є оптимальним. Описи відео можуть містить різноманітну інформацію, а належність до одного чи іншого класу може визначатися лише одним словом з десяти. Системам що базуються на правилах також важко подолати лексичні невизначеності, як, наприклад, ті ж омографи, для передбачення яких розробнику потрібно витратити багато часу.

2) Метод класифікації тексту на основі машинного навчання. Ці системи класифікації працюють, застосовуючи алгоритми аналізу наборів даних на наявність шаблонів, які пов’язані з певним класом. Отримуючи навчальну вибірку, алгоритм аналізує дані на наявність певних шаблонів, які пов’язані з наданими класами. Після навчання модель можна оцінити на точність, перевіривши співпадіння з вхідними класами.

Цей метод був обраний для вирішення задачі генерації ключових слів. Алгоритми машинного навчання можуть досягати великого рівня точності при вирішенні задач класифікації при роботі з великими обсягами інформації та при наявності великої кількості визначених класів. Алгоритми штучного інтелекту також дають можливість врахувати складні шаблони та взаємозв’язки в даних, на відміну від методу що базується на визначених правилах. Також побудовані моделі мають можливість автоматично оновлюватися, навчаючись на нових даних, що забезпечує підтримання їх актуальності.

Одним з конкретних прикладів застосування класифікації текстової інформації є визначення необхідної області медичної консультації, базуючись на описі захворювання або травми. Для вирішення задачі було використано інструменти представлення та нормалізації тексту та модель машинного навчання для отримання ефективної моделі передбачення наявного захворювання людині на забраному анамнезі [3].

Наявні також приклади застосування штучного інтелекту для генерації ключових слів відео за його змістом з використанням попередньої класифікації. Існують різні підходи, що базуються на різних вхідних даних. Це може бути аналіз зображення відео, яке розбивається на послідовність кадрів, які аналізуються на схожість. Після цього відбувається оцінка схожості цих кадрів за допомогою попередньо побудованого вектору з використанням індексу структурної схожості. Це дозволяє обрати кадр, який містить в собі найбільше інформації [4]. Другий варіант – аналіз текстового опису відео, яке аналізується, нормалізується та подається до моделі штучного інтелекту для подальшого навчання. Цей метод буде розглянутий в даній дипломній роботі [5].

Вирішення задачі генерації ключових слів відео за його текстовим описом безпосередньо пов’язана з областю обробки природньої мови. Для вирішення задачі було використано такі інструменти як: токенезація слів, застосування лематизації, визначення та усунення стоп–слів, застосування регулярних виразів для форматування тексту та підходи для його представлення.

1.2 Основні підходи представлення тексту

Алгоритми машинного навчання працюють шляхом аналізу числових даних. Таким чином, перед надання інформації до моделі для навчання попередньо його потрібно перевести в числовий формат. Цей процес називається представленням тексту.

Bag–of–words, або «мішок слів» – підхід до векторизації та представлення текстової інформації, який базується на складанні одного масиву (або вектору), який містить усі слова наданого текстового фрагменту. При цьому не враховується порядок слів або граматика, враховується лише кількість зустрічей унікальних слів в цьому фрагменті. Алгоритм цього методу реалізований наступним чином: на вхід подається текстова інформація у вигляді строк, кількість строк визначає кількість векторів. Наступним кроком є визначення розміру кожного вектору, який залежить від загальної кількості унікальних слів [6]. Наприклад якщо в 2 строках містить 50 унікальних слів, довжина кожного вектору буде сягати 50. На цьому моменті виникає проблема ймовірності отримати надвеликий розмір кожного вектору, у разі якщо вхідні дані також великі і складаються з тисяч унікальних елементів. При цьому отримані вектори будуть мати багато нулів, такі вектори називаються розрідженими і потребують більше пам’яті та часу для обробки. Для уникнення цієї проблеми змінюють параметри інструменту векторизації або застосовують додаткові методи форматування вхідних даних, такі як: ігнорування регістру та пунктуації, видалення стоп слів, застосування лематизації. Після надходження вхідних даних до інструменту векторизації, формується словник унікальних слів, де кожне слово має свій номер в утворених векторах. Наступним кроком є визначення кількості векторів та його розміру, вектори заповнюються 0 або 1, які вказують чи зустрічається слово з словнику в визначеному документі.

Наприклад, узято 4 речення англійською мовою:

It’s funny movie, I like it.

Movie I this hate.

It was awesome. I like this.

I nice one. It love.

Коли усі ці речення передані до інструменту реалізації підходу bag–of–words, з них формується словник унікальних слів. В цьому випадку його розмір

буде становити 11 слів. Після формування словнику, відбувається оцінка слів в кожному з речень (документів), перетворюючи текст на набір чисел в векторі. Якщо слово з словника зустрічається в реченні – ставиться 1, якщо слово з словника в реченні відсутнє – ставиться 0. Результат отриманих векторів для 4 речень наведені в табл. 1.1.

Таблиця 1.1

4 вектори, сформовані за допомогою підходу bag–of–words

№	awesome	funny	hate	it	like	love	movie	nice	one	this	was
0	0	1	0	1	1	0	1	0	0	1	0
1	0	0	1	0	0	0	1	0	0	1	0
2	1	0	0	1	1	0	0	0	0	1	1
3	0	0	0	1	0	1	0	1	1	0	0

Розглянутий підхід реалізує невпорядковане представлення тексту, що приймає до уваги лише кількість зустрічей певного слова, втрачаючи інформацію про їх порядок. Для врахування цієї особливості можна застосувати N–грам, які дозволяють врахувати просторову інформацію. Біграми враховують пари по 2 слова, триграми по 3 і т.д. Підхід bag–of–words є актуальним для вирішення багатьох задач класифікації, системи в основі яких лежить представлення інформації з використанням цього методу демонструють гарні результати точності та швидкості опрацювання великих обсягів даних [7]. Цей метод представлення був використаний для вирішення поставленої задачі через простоту його реалізації, швидку обробку векторів, підготовлених цим методом, гнучкість, яка полягає у швидкій адаптації до різних мов та корпусів даних, можливості застосування оцінок TF–IDF.

TF–IDF – частота терміну, зворотня частоті документу. Даний підхід також створює вектор, який представляє документ, базуючись на словах, що в ньому містяться. На відміну від «мішка слів» ці слова зважені не тільки за фактом їх зустрічі, TF–IDF також оцінює важливість слів в документах шляхом

визначення релевантності слова за частотою його зустрічі в документі, частота зустрічі слова в документі є мірою значимості слова [8]. Кількість слів у векторі замінюється оцінкою TF–IDF, яка розраховується для цього набору даних. Аналогічно до методу bag–of–words спочатку визначається факт зустрічі слова в документі, при цьому обмежується вплив найбільш розповсюджених слів та слів, що не містять інформації про вміст тексту («stop–words»), таких як, наприклад, «або», «не» та ін. Слова, які зустрічаються частіше, відносяться до «зворотньої частоти документа», це зумовлено тим, що чим частіше в текстових наборах даних зустрічається слово, тим менш можливо відрізнити його від інших текстових даних з аналогічними словами.

Формула, яка використовується цим методом:

TF (частота слова) – визначає відношення кількості зустрічей визначеного слова до усієї кількості слів в документі. Обчислюється за формулою (1.1):

$$TF = \frac{n_i}{\sum_k n_k}, \tag{1.1}$$

де n_i – число зустрічі слова в наборі слів, $\sum_k n_k$ – усього слів в документі.

IDF (обернена частота документа) – зворотній показник частоти, з яким визначене слово зустрічається в документах колекції. За допомогою розрахунку цього значення можна зменшити вплив часто зустрічаємих слів. Обчислюється за формулою:

$$IDF = log\frac{|D|}{|(d_i \supset t_i)|}, \tag{1.2}$$

де – кількість документів в колекції, $|(d_i \supset t_i)|$ – кількість документів, в яких зустрічається слово t_i.

Значення TF–IDF – це результат виразу TF * IDF. Таким чином більшу оцінку мають слова з високою частотою зустрічі в документі та незначною частотою зустрічі в інших документах [9].

Приклад застосування міри TF–IDF для формування векторів за текстовими даними див. табл. 1.1 наведено в табл. 1.2

Таблиця 1.2

4 вектори, сформовані за допомогою підходу TF–IDF

№	awesome	funny	hate	it	like	love	movie	nice	one	this	was
0	0.000	0.571	0.000	0.365	0.450	0.000	0.450	0.000	0.000	0.365	0.000
1	0.000	0.000	0.702	0.000	0.000	0.000	0.553	0.000	0.000	0.448	0.000
2	0.539	0.000	0.000	0.344	0.425	0.000	0.000	0.000	0.000	0.344	0.539
3	0.000	0.000	0.000	0.345	0.000	0.541	0.000	0.541	0.541	0.000	0.000

Підхід TF–IDF був використаний при вирішенні задачі генерації ключових слів, тому що він дозволяє зменшити вагу часто вживаних слів, що несуть менше інформації щодо вмісту відео та збільшити вагу слів, які зустрічаються рідше, при цьому даючи більше інформації. Цей підхід також дозволяє виокремити специфічні ключові слова, які допоможуть краще відрізнити різні описи відео між собою, а також зберегти контекст окремих слів у межах документу, що допоможе алгоритму класифікації краще розуміти контекст та вирішувати поставлену задачу з більшою точністю.

Word2vec – це штучна нейронна мережа, яка використовує шари, для обробки тексту та формування векторизованого представлення тексту. Вхідні дані для Word2vec це великий корпус текстової інформації, за допомогою якого будується простір векторів, розмірність котрого залежить від відповідних налаштувань. Кожне унікальне слово представлено у вигляді вектору. Цей метод використовується для перетворення лінгвістичного контексту в числовий формат. Вектори слів відповідно розміщені таким чином, що відстань між ними, в багатовимірному просторі відносно невелика. Це визначає, що схожі за значенням слова будуть розміщені ближче один до одного, ніж різні. Ця модель дає змогу фіксувати синтаксичні та семантичні схожості між словами.

Таким чином, якщо розглянути приклад з 4 словами: man, queen, king, woman, можна одразу визначити, що king має більше схожості зі словом man,

відповідно і розташовані вони будуть ближче один до одного. Аналогічні твердження застосовані для пари queen та woman. Приклад розташування цих слів за їх векторами наведено на рис. 1.1.

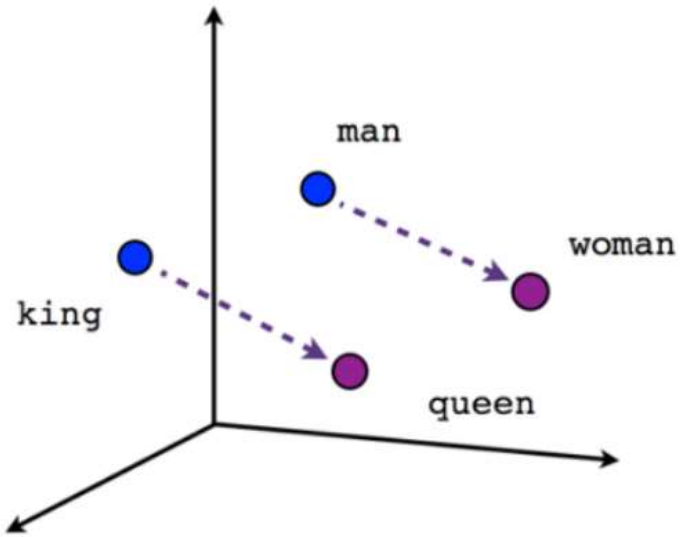

Рисунок 1.1. Вектори слів в багатовимірному просторі

Для утворення корпусу слів використовується два підходи: CBOW, сенс якого полягає у передбаченні ключового слова, використовуючи слова з контексту, Skip–Gram, який навпаки, полягає у передбаченні слів контексту, використовуючи ключове слово. При навчанні однієї з наведених моделей також обов’язково зазначається розмірність вкладень, яка представляє собою кількість вимірів просторів векторів, в які перетворюються слова. Якщо ж визначено, що на вхідних даних навчити модель та сформувати корпус слів буде проблематично, запропоновані попередньо сформовані корпуси слів, які були навчені на великих, підготовлених масивах текстової інформації.

Підхід GloVe – це метод отримання векторних представлень слів, який враховує глобальну статистику співвідношення слів у тексту. Цей підхід націлений на дуже великі корпуси даних. Під час визначення векторів слів,

спочатку розраховується логарифм вірогідності спільної зустрічі слів і та j в текстовому корпусі в середині 10 шарів.

$$Loss = \sum_{ij} f(P_{ij}) * (u_i u_j' + b_i + b_j' - logP_{ij})^2 = min_{u_i u_j' b_i b_j'} \quad (1.3)$$

В процесі навчання моделі підбираються параметри – компоненти векторів та b_i – параметр зміщення, таким чином, аби мінімізувати функціонал похибки (1.3).

Де $f(x)$ визначається як:

(1.4)

У відкритому доступі наявні готові набори векторів, отримані проведенням векторизації на великих корпусах даних. Готові набори векторів містять в собі 400 тисяч слів, а їх розмірність варіюється від 30 до 300. Під час виконання дипломної роботи, одним з розглянутих методів векторизації тексту є GloVe. Для його застосування було використано готовий корпус даних, розмірність вектору якого становить 50, кількість слів в корпусі – 400 тисяч.

Цей метод також був реалізований програмно, оскільки дозволяє краще захоплювати семантичні зв'язки між словами та забезпечує більш багатий контекст для кожного слова, що допоможе підвищити точність класифікації та зробить систему більш ефективною у розрізненні текстових описів відео.

1.3 Алгоритми машинного навчання для класифікації

В даній дипломній роботі, при розробці програмного додатку мовою Python було використано три алгоритми машинного навчання для обробки текстових описів відео – RandomForestClassifier, XGBoost, LinearSVC.

RandomForestClassifier – клас бібліотеки SKLearn мови програмування Python, який використовує метод випадкового лісу як алгоритм машинного навчання.

Метод випадкового лісу використовує ансамбль дерев рішень. Дерево рішень використовується в алгоритмах машинного навчання для отримання одного рішення базуючись на вхідних параметрах. Гілки цього дерева містять в собі ознаки, а листя містить значення, яке визначається. Аби отримати необхідне рішення потрібно спускатися по дереву, визначаючи ознаки. Приклад дерева рішень наведено на рис. 1.2.

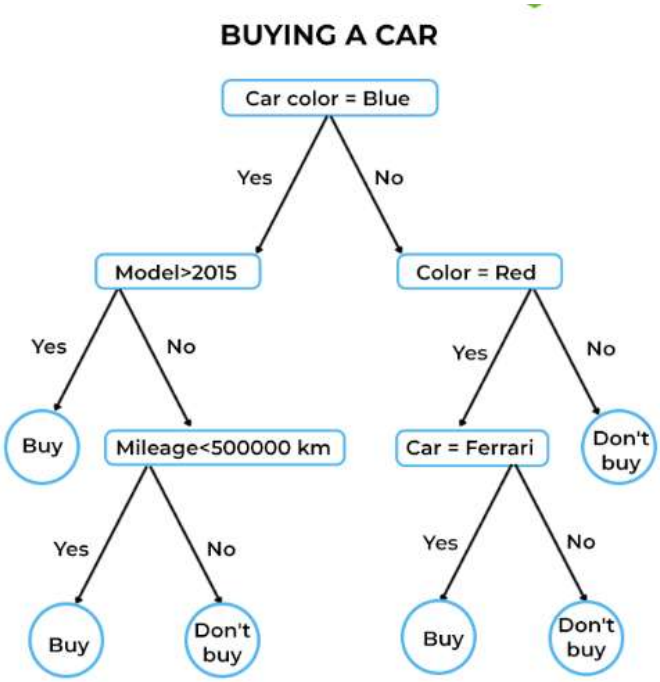

Рисунок 1.2. Приклад дерева рішень

Сутність алгоритму полягає у використанні ансамблю з великою кількістю дерев рішень, кожне з яких саме по собі дає невелику якість класифікації, але за рахунок їх великої кількості результат є прийнятним [10].

Найбільш розповсюджений спосіб побудови дерев ансамбля – бегінг (від англ. bootstrap aggregation) відбувається в 3 етапи:

1. Генерується випадкова повторна підвибірка розміром N з навчальної вибірки. Деякі з елементів попадуть в неї більше одного разу, а отже в середньому $N\left(1-\frac{1}{N}\right)^N$ елементів будуть невідібраними.

2. Будується дерево рішень, яке класифікує елементи створеної підвибірки, обираючи при побудові вузлів набір ознак з тих, на основі яких

відбувається розбиття. Вибір найкращих з цих ознак відбувається з використанням алгоритму Джині.

3. Дерево рішень будується до повного вичерпання підвибірки, при цьому не відбувається відсічення надлишкових гілок.

Класифікація, в свою чергу, відбувається шляхом голосування, де кожне дерево рішень класифікує об’єкт за відповідним класом. Перемагає той клас, за який віддало свій голос більша кількість дерев.

Кількість дерев визначається при отриманні мінімальної помилки класифікатора на тестовій виборці.

Основною перевагою цього методу є можливість ефективної обробки даних з великою кількістю ознак та класів, можливість до масштабування, внутрішня оцінка здібності моделі до узагальнення, вбудовані методи оцінки значимості окремих ознак в моделі [11].

З недоліків можна зазначити великий розмір отриманих моделей. З зростанням кількості дерев лінійно зростає і розмір моделі в пам’яті комп’ютера.

XGBoosting – клас бібліотеки xgboost мови програмування Python, який використовує метод градієнтного бустінгу як алгоритм машинного навчання.

Бустінг являє собою техніку побудови ансамблів моделей, в якому передбачення побудовані послідовно. Основою є ідея, яка полягає у навчанні кожної наступної моделі враховуючи помилку попередньої. Моделі для ансамблю можуть бути різноманітні: дерева рішень, регресійні, класифікаційні. Виходячи з ідеї навчання, базуючись на попередніх помилках, для навчання моделі потребується менше часу.

Градієнтний бустінг – це алгоритм машинного навчання для задач класифікації, яка будує модель передбачення у вигляді ансамблю слабких моделей передбачення.

Алгоритм градієнтного бустінгу також покладається на визначення оптимального рішення шляхом збалансованого голосування. Модель цього голосування представлена формулою (1.5)

, (1.5)

де X – простір, з якого обираються об'єкти $b_i, a_i,$ – коефіцієнт перед моделлю, та безпосередньо сама модель. Можна припустити, що на певному кроці за допомогою правила, описаного (1.5) вдалося додати до композиції $i-1$ слабку модель. Для визначення конкретної моделі на певному кроці використовується функція похибки.

(1.6)

Виходячи з цього, найкраща модель та, яка найсильніше зменшує похибку, отриману на попередніх ітераціях. Так як бустінг градієнтний, функція має вектор антиградієнту, вздовж якого відбувається пошук мінімуму. Вектор антиградієнту знаходиться за формулою (1.7)

(1.7)

Таким чином, вектор антиградієнту буде дорівнювати . На кожному кроці i рахується похибка моделі, отриману на попередніх ітераціях. Наступним кроком нова модель навчається на цих помилках, а отриманий результат перетворюється на від'ємний, додається коефіцієнт і отриманий результат додається до ансамблю.

При використанні XGBoost, який використовує в якості моделі дерева пошуку рішень, до функціоналу похибки додається регуляризація.

$$err(h) = \sum_{j=1}^{N} L \tag{1.8}$$

Кожне дерево рішень штрафується, якщо сума норм значень на його листі велика. Таким чином складність дерева описується формулою:

, (1.9)

де – значення на листі дерева, – кількість дерев.

Нове розбиття обирається таким чином, аби максимізувати Gain:

$$Gain = \frac{G_l^2}{S_l^2+\lambda} + \frac{G_r^2}{S_r^2+\lambda} - \frac{(G_l+G_r)^2}{S_l^2+S_r^2+\lambda} - \gamma, \quad (1.9)$$

де λ, γ – числові параметри регуляризації, – відповідні суми з перших та других похідних при даному розбиттю.

У підсумку можна зазначити, що XGBoost має високу точність та швидкодію, через використання апаратної оптимізації та паралелізації. В цьому алгоритмі наявна регуляризація, яка дозволяє уникнути перенавчання, при використанні складних моделей. Алгоритм може показувати гарний результат як при роботі з розрідженими датасетами за рахунок заповнення пропущених значень в залежності від значень втрат так і при роботі з збалансованим датасетом за допомогою використання методу зважених квантилей, який дозволяє ефективно знаходити оптимальні точки розділення.

Одним з недоліків цього алгоритму є чутливість до гіперпараметрів, що визначає необхідність у ретельному налаштуванні цих параметрів перед навчанням моделі. Незважаючи на наявну регуляризацію, неправильно визначені гіперпараметри можуть привести до перенавчення або недостатньої навченості моделі.

LinearSVC – це модель машинного навчання, яка використовується опорні вектори для вирішення задач класифікації. Він являє собою один з варіантів методу опорних векторів, який використовує лінійну функцію розділення класів у просторі ознак.

Основна ідея методу полягає у знаходженні гіперплощини, яка дає можливість оптимально розділити дані на класи. Ця гіперплощина будується

таким чином, аби максимізувати відстань між нею та найближчими точками кожного класу, які називається опорні вектори.

Формула для роздільної площини у випадку лінійної моделі LinearSVC має вигляд:

, (1.10)

де – функція роздільної площини, – вектор ваг, – вхідні дані, – зміщення.

Головна мета алгоритму – знайти такі оптимальні значення , аби мати змогу розділяти дані з якомога більшим проміжком. Це досягається за рахунок мінімізації функції втрат, яка вираховує помилку класифікації та ширину маржину. Зазвичай функція втрат виглядає наступним чином:

, (1.11)

де – квадрат евклідової норми ваг, – параметр регуляризації, який контролює компроміс між розділяючою гіперплощиною і кількістю помилок, y_i – мітка класу для i–го прикладу даних, x_i – безпосередньо i–тий приклад даних.

Перевагами цього алгоритму є висока ефективність роботи з великими обсягами вхідних даних, оптимальне використання пам’яті, за рахунок застосування підмножини навчальних точок в функції прийняття рішень [12]. З недоліків можна виділити чутливість до шуму, за наявності якого у вхідних даних отримана модель може виконувати помилкову класифікацію, особливо якщо класи не можуть бути чітко розділені лінійною гіперплощиною.

1.4 Засоби оцінки якості моделі

У машинному навчанні метрика оцінки продуктивності відіграє дуже важливу роль у визначенні продуктивності отриманої моделі машинного навчання для набору даних, з яким вона ніколи не контактувала. Враховуючи,

що навчені моделі використовуються для вирішення задач в умовах постійного надходження нових даних, які потребують класифікації, важливо чітко визначити показники, які свідчать про якість моделі та її можливість узагальнювати. Найімовірніше, навчена модель, завжди буде краще працювати з набором даних, але цей показник точності не є абсолютним, таким чином виникає необхідність в інших метриках оцінки продуктивності. Метрика оцінки продуктивності визначає, чи буде навчена модель машинного навчання добре справлятися з вирішенням проблеми, для якої вона була навчена, чи ні.

Існує багато показників оцінки продуктивності, які можна використовувати для оцінки якості моделей машинного навчання для класифікації, а також для регресії.

Генерація ключових слів відео за його описом включає в себе класифікацію багатьох класів, що означає розподіл n екземплярів між $k \geq 3$ класами. Перед переходом до розгляду метрик, необхідно визначити концепцію опису цих метрик в термінах похибки класифікації – confussion matrix, або матриця помилок.

Легше усвідомити алгоритм роботи матриці помилок, розглядаючи приклад бінарної класифікації. Нехай існує 2 класи, між якими модель має класифікувати 1000 об’єктів. Результат роботи моделі представлений у вигляді матриці помилок в табл. 1.3.

Таблиця 1.3

Матриця помилок для моделі бінарної класифікації

	y_1	y_2
$\hat{y}_1$	750 (TP)	25 (FP)
$\hat{y}_2$	75 (FN)	150 (TN)

Враховуючи, що , y_2 – це істинні значення тестової вибірки, а $\hat{y}_1$, $\hat{y}_2$ – це припущення, зроблені навченою моделлю, можна розділити отримані значення на 4 категорії, які представлені на рис. 1.3:

1) TP – true positive. В цьому випадку класифікатор вірно визначає клас об’єкту, що розглядається

2) TN – true negative. В цьому випадку класифікатор вірно визначає, що об’єкт не належить класу, що розглядається

3) FP – false positive. В цьому випадку класифікатор неправильно відніс об’єкт класу, що розглядається

4) FN – false negative. В цьому випадку класифікатор неправильно визначає, що об’єкт не належить класу, що розглядається.

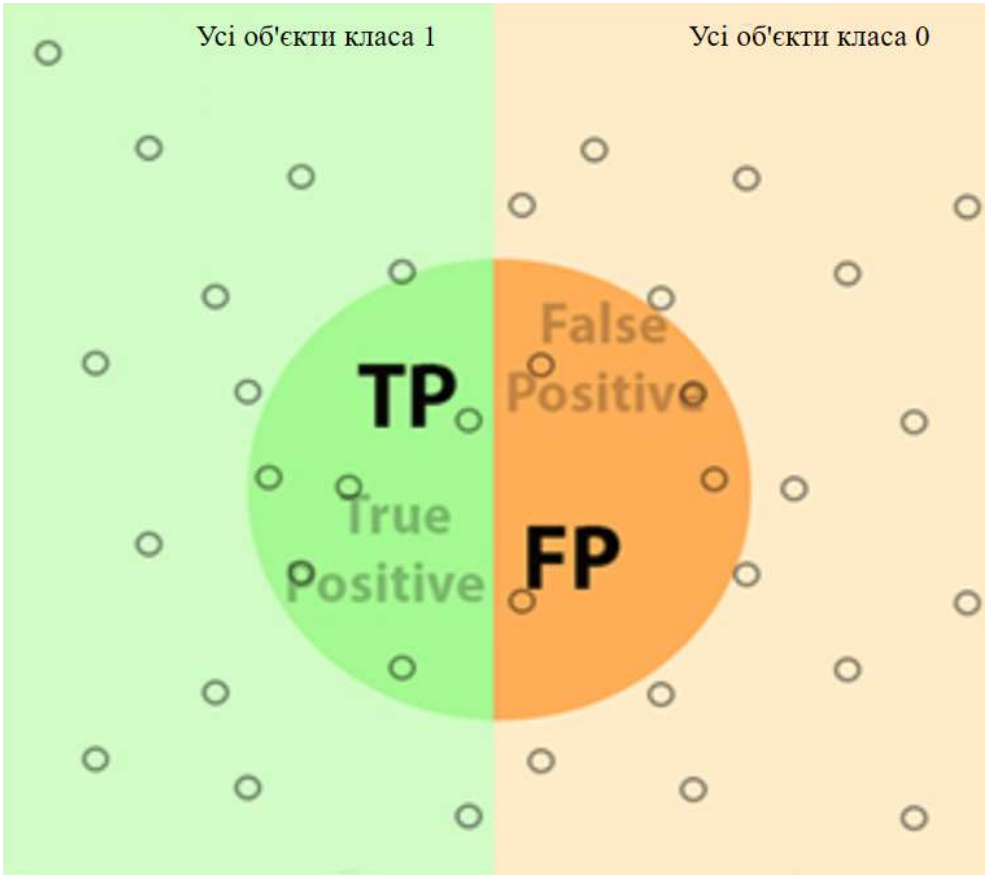

Рисунок 1.3. Візуалізація оцінки передбачення моделі

Перед розглядом метрик, необхідно розглянути приклад матриці помилок для класифікації багатьох класів, яка буде реалізована в цій дипломній роботі. Типова матриця помилок для класифікації багатьох класів наведена в табл. 1.4.

Таблиця 1.4

Матриця помилок для моделі багатокласової класифікації

	y_1		y_3
$\hat{y}_1$	T_1	F_{12}	
$\hat{y}_2$	F_{21}	T_2	F_{23}
	F_{31}	F_{32}	T_3

При розгляді багатокласової класифікації значення TP, TN, FP, TN будуть розраховані відносно класу i за формулами (1.12–1.15).

(1.12)

$$FP_i = \sum_{c \epsilon Classes} F_{i,c} \quad (1.13)$$

$$FN_i = \sum_{c \epsilon Classes} F_{c,i} \quad (1.14)$$

$$TN_i = All - TP_i - FP_i - FN_i \quad (1.15)$$

Найбільш інтуїтивно зрозумілою, але найменш показовою є метрика точності, яка відображає кількість правильних передбачень моделі, відносно загального їх числа.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1.12)$$

Ця метрика якості моделі не є показовою у випадку, якщо класи, визначені моделлю не є збалансованими [13]. Наприклад, нехай, модель визначає позитивне забарвлення відео за описом і може класифікувати його за двома значеннями – 0 – негативно забарвлене, 1 – позитивно забарвлене. З 100 позитивно забарвлених відео, 90 були вірно визначені як позитивно забарвлені, а з 20 негативно забарвлених, 10 були виділені вірно як негативно забарвлені. Таким чином метрика буде дорівнювати:

$$accuracy = \frac{10+90}{10+90+10+10} = 0{,}83$$

В той же час, якщо модель буде просто визначати усі відео як позитивно забарвлені, метрика точності буде мати ідентичне значення.

$$accuracy = \frac{100+0}{100+0+20+0} = 0{,}83$$

При цьому у моделі може буде повністю відсутня можливість узагальнювати інформацію. Для вирішення цієї проблеми необхідно перейти від загальної для всіх класів метрики до окремих показників за кожним класом.

Першою з таких метрик є $precision$. Цю метрику можна інтерпретувати як долю об’єктів, які були передбачені вірно, відносно фактичного класу цих об’єктів.

$$Precision = \frac{TP}{TP+FP} \tag{1.12}$$

Введення метрики $precision$ дозволяє уникнути випадків невірної оцінки моделі, коли більшість об’єктів виявляються записаними до одного класу через зростання значення FP.

Друга метрика, $recall$, визначає яку долю з усіх фактичних класів модель передбачила правильно для кожного з наявних класів.

$$Recall = \frac{TP}{TP+FP} \tag{1.13}$$

Якщо повернутися до візуалізації термінів бінарного результату моделей (див. рис. 1.3.) визначення метрик $precision$та$recall$ може бути зображено на рис. 1.4:

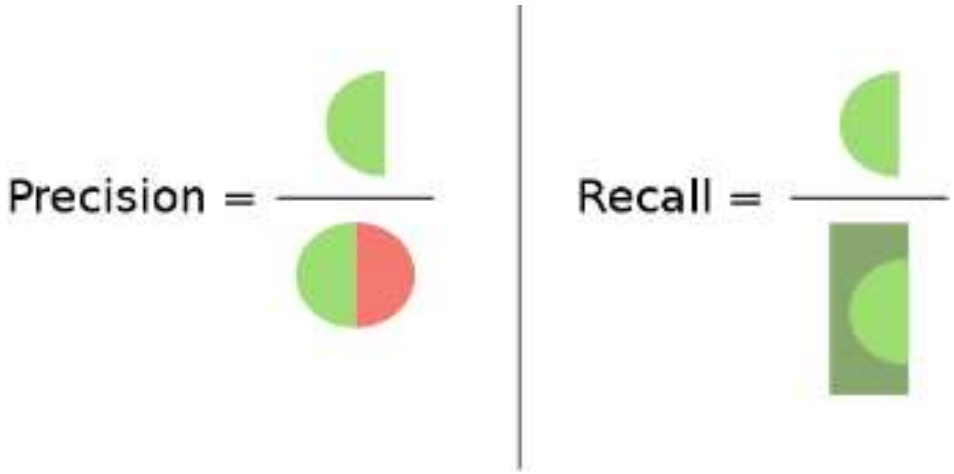

Рисунок 1.4. Принцип роботи метрик precision та recall

Таким чином, recall дозволяє демонструвати здатність моделі відрізнити один клас з поміж інших, а precision – відрізняти клас від інших наявних класів. Ці дві метрики, на відміну від точності не залежать від відношення класів, а отже можуть бути застосовані за наявності незбалансованої вибірки.

Існує також спосіб об'єднати ці два критерія в один агрегований критерій якості, який має назву $F-$критерій. $F-$критерій – це середнє гармонійне між precision та recall, яке розраховується за формулою:

$$F_\beta = (1+\beta^2)\frac{precision*recall}{(\beta^2*precision)+recall}, \quad (1.14)$$

де β – показник, який визначає вагу точності в метриці. Якщо $\beta = 1$, це середнє гармонійне. F–міра досягає максимуму, коли precision та recall дорівнюють одиниці і міра близька до 0, якщо хоча б один з аргументів близький до 0.

Таким чином, за допомогою цих метрик можна оцінити якість навченої моделі, визначити її слабкі та сильні сторона за кожним з класів, що є важливим при вирішенні задач багатокласової класифікації. При налаштуванні гіперпараметрів моделі та покращенні її точності, F–критерій дозволяє сфокусуватися саме на покращенні загального результату якості, а не оптимізації конкретних метрик, а також змістити фокус на вхідні дані.

1.5 Висновки до розділу 1

В даному розділі дипломної роботи було розглянуто сенс та ключовий зв’язок задач NLP з задачами класифікації текстової інформації за допомогою штучного інтелекту, до яких належить також задача генерації ключових слів відео за його описом. Розглянуто принцип задач класифікації тексту, визначено їх практичну застосовність та актуальність. Розглянуто також наявні практичні застосування рішення задач класифікації текстової інформації, наведені в наукових публікаціях.

Описано обов’язковий процес попередньої обробки текстової інформації – представлення, для подальшої можливості використання алгоритмів машинного навчання. Розглянуто методи представлення тексту, які будуть застосовані при подальшій розробці програмного додатку, визначені їх особливості застосування, переваги та недоліки.

Продемонстровано принцип роботи алгоритмів машинного навчання, які найчастіше застосовуються для вирішення задач класифікації. Наведено математичні алгоритми, на яких вони базуються, а також сильні та слабкі сторони.

Зазначено необхідні метрики, які застосовуються для оцінки якості моделей, отриманих за допомогою алгоритмів машинного навчання. Описано принцип їх розрахунку та описові можливості.

2. РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ ДЛЯ ГЕНЕРАЦІЇ КЛЮЧОВИХ СЛІВ ЗА ОПИСОМ ВІДЕО

2.1 Аналіз вхідних даних та формування стратегії вирішення задачі

Для навчання моделі, що генерує ключові слова за текстовим описом відео необхідні вхідні дані. Ці данні безпосередньо впливають на формування алгоритму вирішення задачі щодо генерації ключових слів відео за його описом. Так як поставлена задача безпосередньо є задачею класифікації, необхідно визначити за якими саме класами вона буде здійснюватися і яким чином будуть генеруватися ключові слова. При цьому необхідно враховувати структуру набору даних та його складники, покладаючись на які, можна сформувати оптимальну стратегію вирішення задачі.

Було попередньо визначено, що реалізований додаток буде обробляти та генерувати текстову інформацію англійською мовою, відповідно обраний датасет визначає собою усі необхідні дані, записані англійською.

Датасет був завантажений з джерела, що збирає дані за відео, які потрапили в тренди YouTube з 2020 року [14]. Набір даних має назву US_youtube_trending_data, його формат – cvs. Цей датасет містить дані за усі відео, які потрапили до трендів YouTube протягом 08.2020 – 03.2024 в США. Набір даних сформовано, використовуючи YouTube API. В ньому міститься інформація про: айді відео, його заголовок, дату публікації, айді каналу, назву каналу, айді категорії, дату потрапляння в тренди, теги, кількість переглядів, кількість лайків, дизлайків, коментарів та його опис.

В якості середовища для розробки програмного додатку мовою Python було обрано PyCharm Community Edition. За допомогою мови Python був проведений поглиблений аналіз вхідних даних.

На початку, було додано бібліотеку pandas для роботи з наборами даних:

```
import pandas as pd
```

Наступний крок – підключення даних, здійснюється за допомогою підключеної бібліотеки:

df = pd.read_csv('US_youtube_trending_data.csv')

Перед роботою з набором даних, проведено початкову очистку – видалено рядки, в яких відсутній опис відео.

df["description"] = df["description"].fillna(value="")

Після підключення даних, був проведений початковий аналіз:

print(df.info())

print(df.describe())

В результаті отримано інформацію, яка відображає кількість рядків, яким типам даних вони відповідають, кількість пам’яті що використовується для зберігання датасету та аналіз кількісних показників переглядів, лайків, дизлайків та коментарів. Отримана інформація наведена в табл. 2.1.

Таблиця 2.1

Результат аналізу набору даних з використанням мови Python

№	Стовбець	Кількість строк	Тип даних
1	video_id	263587	object
2	title	263587	object
3	publishedAt	263587	object
4	channelId	263587	object
5	channelTitle	263587	int64
6	categoryId	263587	object
7	trending_date	263587	object
8	tags	263587	object
9	view_count	263587	int64
10	likes	263587	int64
11	dislikes	263587	int64
12	comment_count	263587	int64
13	thumbnail_link	263587	object
14	comments_disabled	263587	bool
15	ratings_disabled	263587	bool

16	description	263587	object

Отже, за отриманими даними визначено, що набір даних містить 263 587 заповнених рядків, кожен з яких містить інформацію про певне відео, яке потрапило до трендів YouTube за період з 2021 по 2024 роки.

Наступним кроком, було проведено аналіз розподілу усіх відео за категоріями [15]. Стовбець category_id містить лише айді категорії. Додатково було сформовано словник, в якому ключ – це номер категорії, значення – його фактична категорія, сформований словник наведено в табл. 2.2.

Таблиця 2.2

Словник для визначення фактичної категорії за її ID

ID категорії	Фактична категорія
1	Film&Animation
2	Auto&Vehicles
10	Music
15	Pets&Animals
17	Sports
19	Travel&Events
20	Gaming
22	People&Blogs
23	Comedy
24	Entertainment
25	News&Politics
26	Howto&Style
27	Education
28	Science&Technology

На першому етапі було сформовано новий датасет, який відображає кількість потрапляння кожної категорії в загальному наборі даних:

category_counts=df['categoryId'].value_counts().to_frame().reset_index().rename(columns={'categoryId': "Категорія", "count": "Кількість"})

Також отримано значення перших п’яти категорій, які зустрічаються найчастіше, наведені в табл. 2.3.

Таблиця 2.3

Числовий показник п’яти найчастіше визначених категорій

Категорія	Кількість
24	52907
20	52278
10	42543
17	30729
22	22251

Застосовуючи бібліотеку matplotlib була реалізована візуалізація отриманого набору даних, представлена на рис. 2.1. Разом з цим, застосований попередньо створений словник категорій.

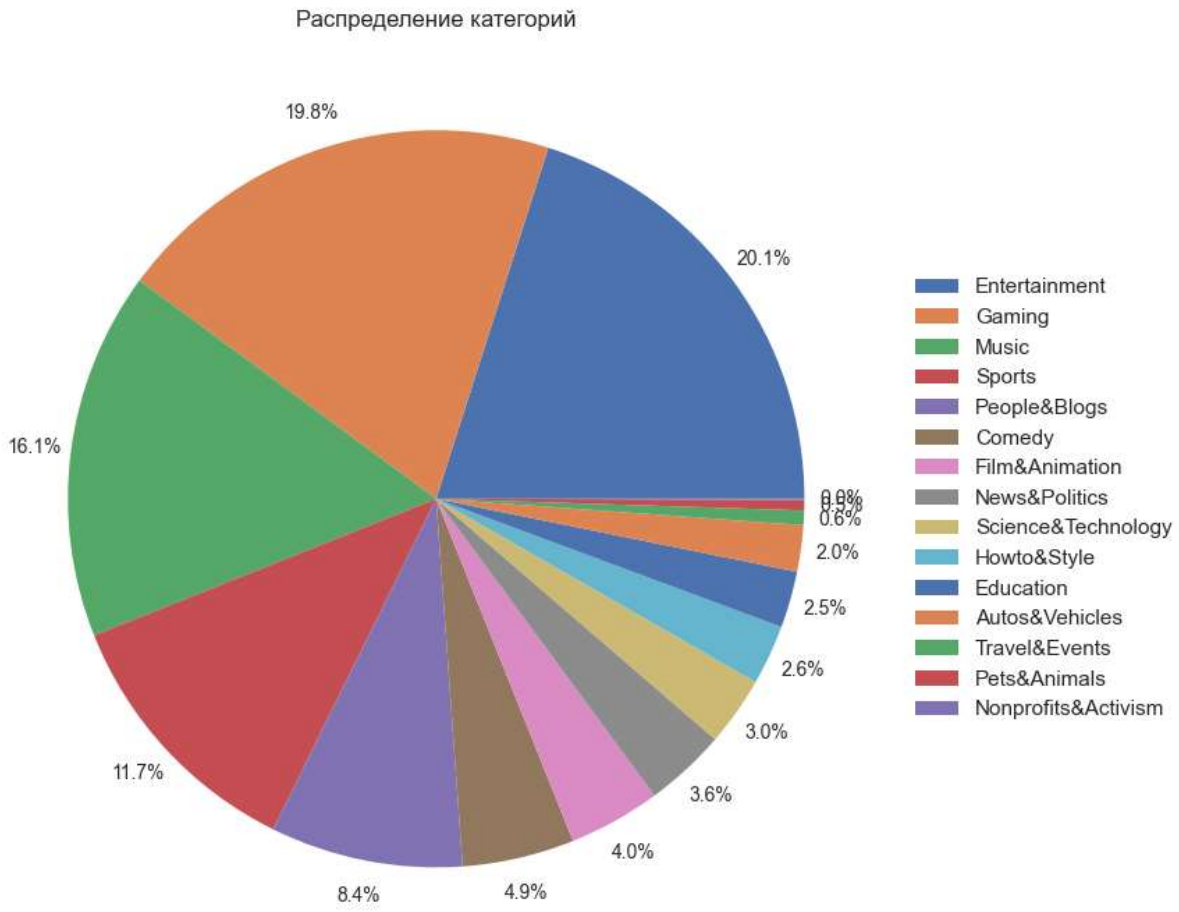

Рисунок 2.1. Загальний розподіл категорій у %

Діаграма відображає нерівномірний розподіл категорій з вираженим домінуванням п’яти категорій над іншими дев’яти, наявними у загальній вибірці.

Врахування частоти категорій важливо при навчанні моделі та формуванні стратегії до генерації ключових слів за описом відео: через перенасиченість певною категорією навчального набору даних можливо нерівномірне її навчання що призведе до хибних передбачень, у разі якщо класом для описів було обрано саме категорію. Перший з варіантів стратегії генерації ключових слів базувався на визначенні 3 можливих категорій, до яких можна було б віднести відео і формуванні з них ключових слів. Після дослідження розподілу категорій цю ідею було відкинуто, тому що, враховуючи мануальний спосіб анотування відео за категоріями, важливо враховувати випадки, коли категорія відео не відповідає, або лише частково відповідає його вмісту. Це може статися якщо користувач, який завантажував відео, обрав категорію яка не відповідає вмісту, але в теорії може охопити більше цільової аудиторії. Визначена особливість обумовлює неможливість формування навчальної вибірки лише за цільовою категорією, яка стосується запланованої тематики. У випадку, коли поставлена задача не полягала у класифікації відео певної тематики, проблему можна було б вирішити балансуванням вибірки перед її розподілом на навчальну та тестову.

Наступним варіантом було розглянуто підхід генерації ключових слів, який базувався на визначенні переліку тегів з набору даних як класу для кожного з описів. Було проведено аналіз тегів які найчастіше зустрічаються в кожній категорії та загалом. Для цього в кожному рядку було проведене форматування: значення в кожному рядку було розбито на масив окремих тегів та приведено до нижнього регістру. За допомогою перебору було розраховано кількість найчастіше вживаних тегів загалом та за 3 категоріями. Розрахунки було візуалізовано та подано у вигляді діаграм (рис. 2.2 – 2.5).

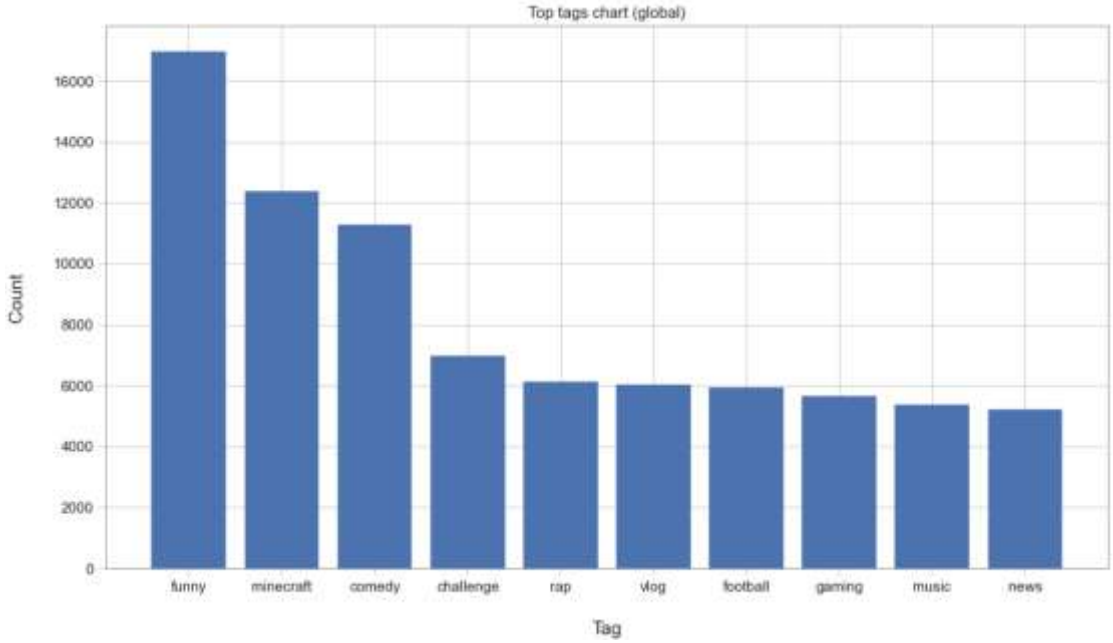

Рисунок 2.2. Глобальний розподіл 10 найчастіше вживаних тегів за популярністю

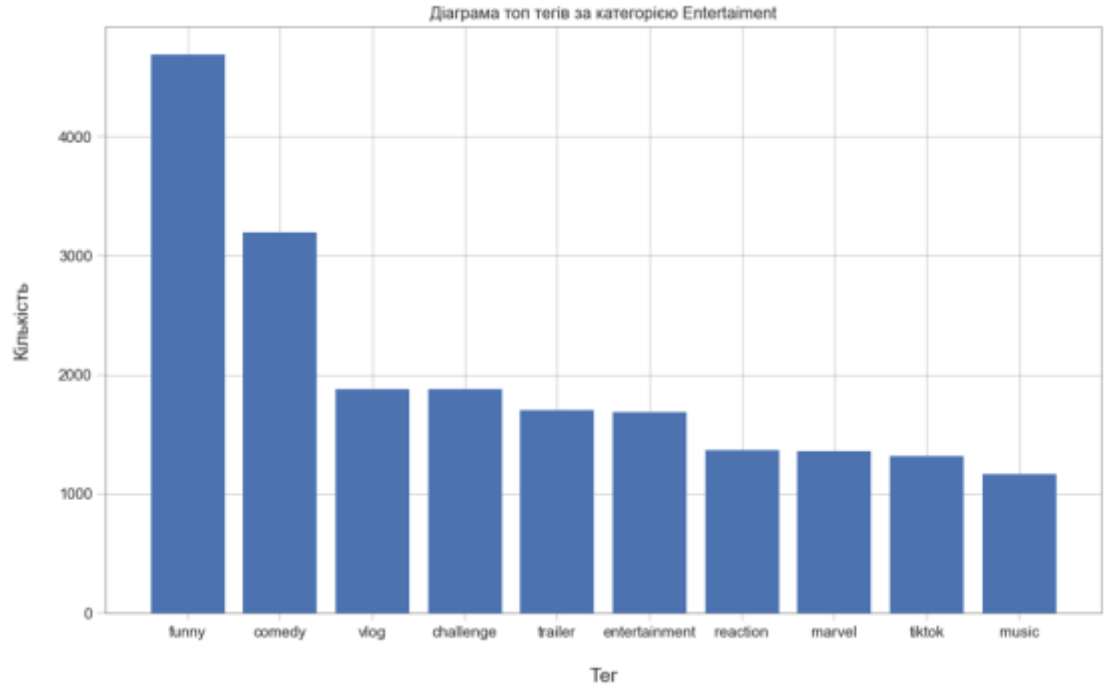

Рисунок 2.3. Розподіл 10 найчастіше вживаних тегів за популярністю в категорії Entertainment

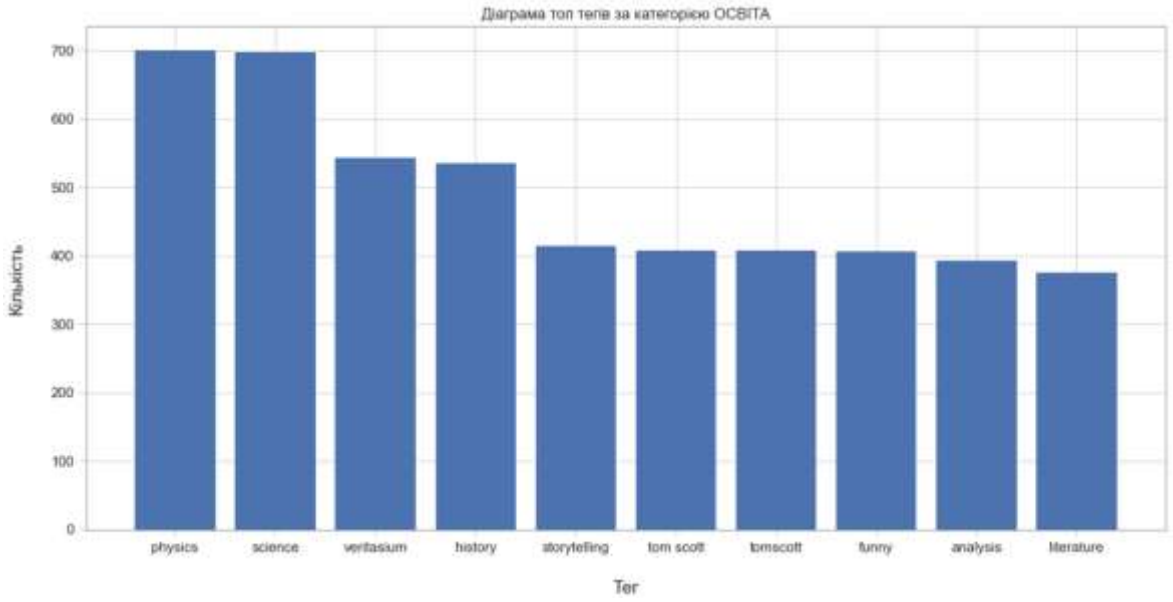

Рисунок 2.4. Розподіл 10 найчастіше вживаних тегів за популярністю в категорії Education

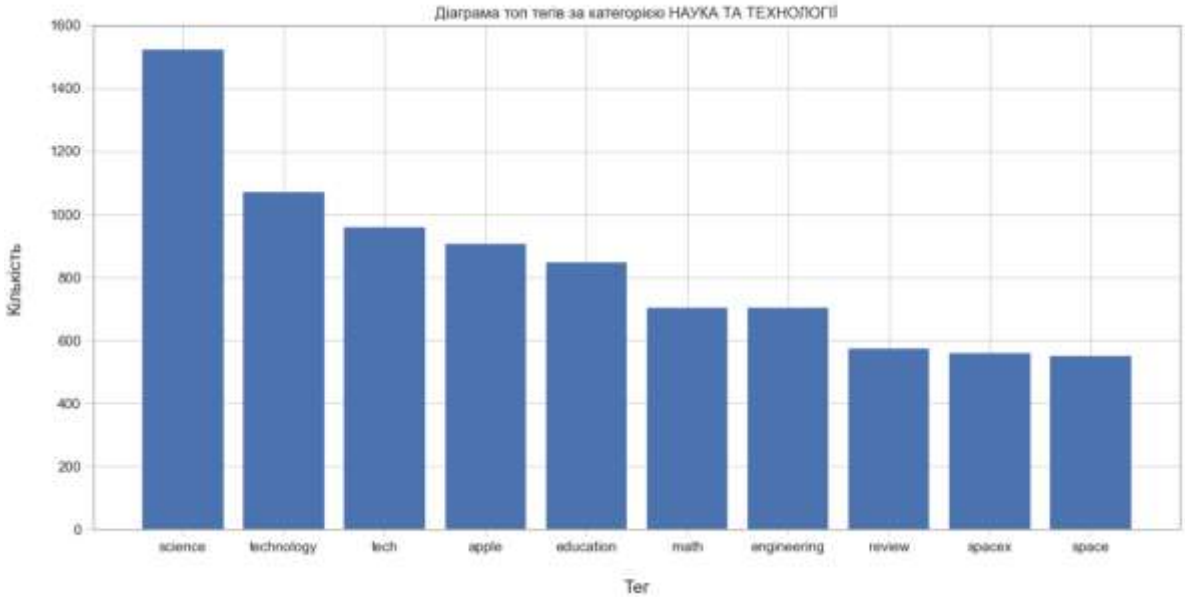

Рисунок 2.5. Розподіл 10 найчастіше вживаних тегів за популярністю в категорії Science&Technology

За результатами аналізу можна простежити, що кожна категорія містить в собі 3–4 ключові слова, які за популярністю переважають інші. Також було визначено, що розподіл тегів є довільним для кожного з відео через мануальну процедуру їх зазначення на платформі YouTube. Яскравим прикладом є наявність тегу funny, як одного з найчастіше вживаних в відео з зазначеною категорією «Освіта». Цей тег не несе інформаційного навантаження щодо вмісту відео і ніяким чином не може описати його тематику, але неможливо

передбачити скільки подібних тегів без інформаційного навантаження зустрінуться в інших відео. Отже, використання наборів тегів як класів для описів також не є коректним через їх високу різноманітність та можливу недостовірність щодо вмісту відео. В той же час, дослідження популярних тегів за категоріями допомогло сформулювати певну базу ключових слів, розділених на 3 категорії.

Наступним кроком було досліджено зміну популярності 4 найуживаніших актуальних тегів за часом. Це допомогло б довести недоцільність використання тегів, або наборів тегів як класів. Зміну частоти зустрічі 4 потенційно найпопулярніших тегів за визначеною тематикою наведено на рис. 2.6.

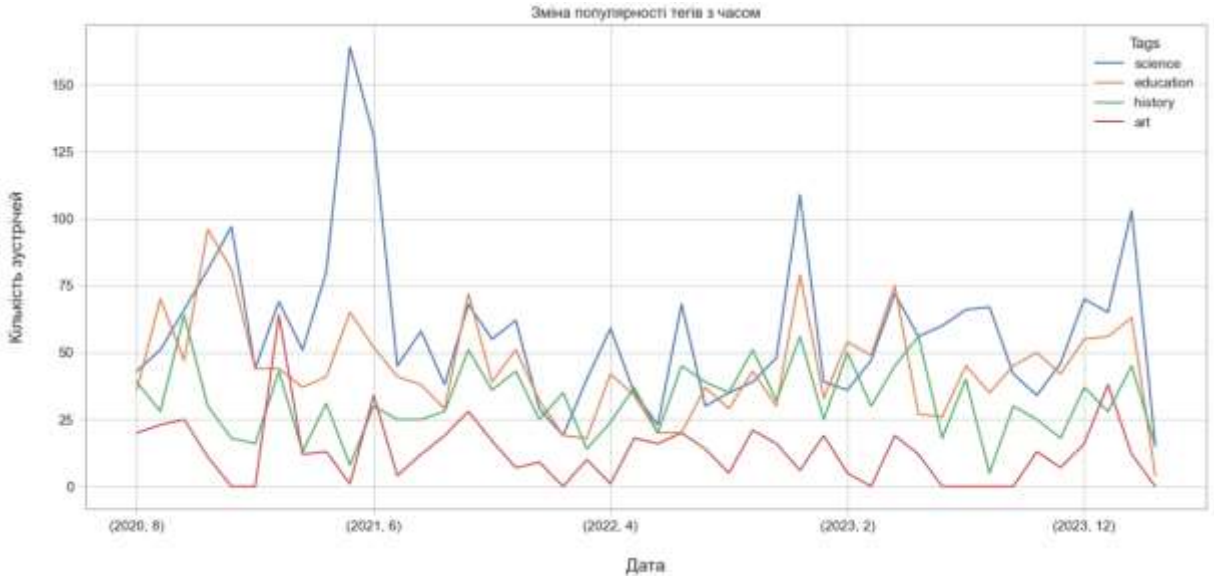

Рисунок 2.6 Зміна частоти зустрічі 4 визначених тегів за часом

За графіком можна простежити, що популярність зазначення тегів science, education, history, art має незначні зміни протягом визначеного часового періоду. Ці зміни є невпливовими, попри існування невеликих коливань, які не мають суттєвого впливу на навчання моделі. Отже, проведений аналіз унеможливлює розгляд варіанту фільтрування вибірки за певним часовим періодом, який би охоплював момент одночасної популярності усіх необхідних тегів, які могли б бути використані в якості класів. Попри це, також відсутня суттєва зміна популярності найчастіше вживаних тегів, що могли б

зустрічатися у відео на різні тематики являючи собою шумові перешкоди для моделі штучного інтелекту.

Провівши аналіз вхідних даних, було сформовано стратегію вирішення задачі, яка полягає у здійсненні попереднього автоматичного анотування за наявністю необхідних ключових слів у тегах відео. В таблиці 2.4 наведено сформовані 3 набори ключових слів, по 10 слів у кожному.

Таблиця 2.4

3 групи ключових слів за номерами

№	Перша група	Друга група	Третя група
0	science	education	history
1	discovery	politics	art
2	technology	geography	documentary
3	engineering	law	mystery
4	review	tutorial	film
5	invention	math	animation
6	tech	economics	cartoon
7	software	physics	news
8	pc	social	stories
9	space	investing	design

Ці 3 групи слів дозволяють б визначати ключову тематику відео, його направленість та спосіб подання. За кожною з груп було сформовано словник, який дозволяє передати ключове слово його номером.

Кожний вміст стовбця tags для кожного відео був проаналізований на вміст слів з 3 зазначених (див. табл. 2.4) груп. Проведення автоматичного анотування розширило кількість стовбців набору даних на 3. Таким чином, сформований набір даних є оптимальним та містить необхідну кількість даних для створення моделі класифікації яка вирішує задачу генерації ключових слів відео на основі його опису. В таблиці 2.5 наведено кількість відео за кожним з словом з трьох груп.

Таблиця 2.5

Кількість відео, в яких зустрічаються слова з кожної групи ключових слів

№ слова	Перший набір слів	Другий набір слів	Третій набір слів
0	296	255	243
1	88	253	112
2	137	69	166

3	171	23	147
4	585	478	257
5	56	30	675
6	202	29	182
7	63	221	1442
8	225	61	75
9	250	52	75

Проаналізувавши кількісні показники, було виявлено, що їх достатньо для отримання адекватного результату при утворенні моделі алгоритмом машинного навчання, тому що кількість елементів кожного класу становить більше 50. В середньому, за кожною з групою слів було обрано по 2000 відео, за якими попередньо була здійснена автоматична анотація. Повний лістинг коду, за допомогою якого здійснено аналіз вхідних даних наведено у ДОДАТКУ Г. Лістинг коду, за допомогою якого проведено автоматичну анотацію, наведено в ДОДАТКУ Д.

2.2 Форматування та представлення даних

Перед представленням тексту та передачу його до моделі, його необхідно форматувати. Для цього була створена відповідна функція:

```
def text_format(text):
    text = text.lower()
    text = re.sub(r'[^a–zA–Za–яA–Я\s]', '', text)
    text = re.sub(r'\s+[a–zA–Z]\s+', ' ', text)
    text = re.sub(r'\^[a–zA–Z]\s+', ' ', text)
    text = re.sub(r'\s+', ' ', text, flags=re.I)
    text = re.sub(r'^b\s+', '', text)
    text = ' '.join(text.split())
    text = ' '.join(word for word in text.split() if not word.startswith(('http', 'instagram', 'twitter')))
    lemmatized_text = ' '.join([lemmatizer.lemmatize(word) for word in text.split()])
```

```
    return lemmatized_text
```

Функція text_format дозволяє здійснити:

1) Приведення всього тексту до нижнього реєстру
2) Видалення спеціальних символів
3) Видалення одиничних символів
4) Видалення подвійних пробілів
5) У разі зустрічі даних, записаних у байтовому форматі, видаляється стартовий символ b
6) Текстова інформація розбивається на окремі слова
7) В кожному наборі слів також видаляються слова, що починаються з https, instagram, twitter (в описі відео дуже часто зустрічаються посилання на соціальні мережі, які не несуть інформації про сам опис)
8) Виконується лематизація даних

Лематизація – процес приведення слів до леми, їх нормальної форми. Таким чином кожне слово приводиться до його кореневої форми, наприклад «біжав» → «біжать». За допомогою лематизації можна уникнути створення функцій, які семантично схожі, але синтаксично відрізняються. Для здійснення лематизації було використано бібліотеку ntlk та метод WordNetLemmatizer.

Описано вище функцію форматування текстових даних було застосовано до описів відео:

```
df['title'] = df[' title '].apply(text_format)
```

Додатково було створену функцію, яка відповідає за виключення «стоп–слів» з вибраних даних. Спочатку їх було завантажено до змінної:

```
nltk.download('stopwords')
nltk.download('punkt')
stop_words = set(stopwords.words('english'))
```

Наступник кроком створено метод, який відповідає за застосування фільтру «стоп–слів».

```
def remove_stopwords(text):
word_tokens = word_tokenize(text)
```

filtered_text = [word for word in word_tokens if word.lower() not in stop_words]

return ' '.join(filtered_text)

Цю функцію, аналогічно, було застосовано до набору описів відео:

df['title'] = df[' title '].apply(remove_stopwords)

Вигляд 10 відформатованих описів відео з застосуванням описаних методів наведено на рис. 2.7.

```
0    horror fan movie awesome watch final trailer s...
1    stalk ig swoozieftwtwitter banditthumbnail zeu...
2    unbearable weight massive talent theater april...
3    uncover past protect future watch new internat...
4    watch demon slayer kimetsu yaiba crunchyroll g...
5    markkermode simonmayo filmreviewsa housewife l...
6    second channel like also keep daily livestream...
7    lord fallen wehypeyjoelhaveranimated alexmarte...
8    realising nt dream job word montell fish video...
9    spooky manor big revealsyou wear s hat murderd...
Name: description, dtype: object
```

Рисунок 2.7 10 відформатованих описів відео

Після здійснення форматування даних, можна перейти до їх представлення, для здійснення якого були використані 3 методи: Bag–of–Words, TF–IDF, Embeddings.

Для представлення тексту за допомогою методу Bag–of–Words було використано клас CountVectorizer бібліотеки sklearn.

Клас CountVectorizer – є простим та ефективним інструментом представлення текстової інформації, за допомогою мови програмування Python. Цей метод приймає в себе текстові дані та набір параметрів, токенезує текстові дані (переводить в список слів) та будує словник слів, що містяться в наборі даних. Після чого повертає текстові дані у векторизованому форматі (зазвичай при середніх–великих розмірах, оптимально повертати результат у вигляді розрідженої матриці).

Під час реалізації, в методі CountVectorizer можна змінювати параметри, які повпливають на інформаційну наповненість сформованого словника:

Параметр stop_words дозволяє відсікти «стоп–слова», які не містять інформації щодо розглянутого тексту. Наприклад вказаний параметр stop–words=«english» прибере такі слова як «the», «a», «I», «or», та ін.

Параметр max_df дозволяє не враховувати слова, які зустрічаються надто часто. Встановивши значення max_df=5, метод не буде враховувати слова, які зустрічаються частіше, аніж в 5 документах.

Параметр min_df дозволяє врахувати слова, які зустрічаються надто рідко. Встановивши значення параметру min_df=3, будуть враховуватися слова, які потрапили мінімум до 2 строк.

Параметр max_features визначає кількість слів, частота зустрічі яких найбільша, що потраплять до кінцевого вектору. Встановивши значення параметру max_features=50, означає що загальний словник буде містити 50 слів.

Ці параметри дають можливість врахувати розмір текстової вибірки та визначити кількість та частоту зустрічі обраних слів в інших строках.

Попередньо інформація з необхідних стовбців була записана до проміжних змінних. Після чого здійснена векторизація з послідуючим формуванням словника термінів.

```
texts = df['title']
cv_descr = CountVectorizer(max_features=1000)
res = cv_descr.fit_transform(texts)
X = res.toarray()
```

За допомогою методу shape, було отримано розмір отриманого масиву, а за допомогою vocabulary_ – вигляд отриманого словнику.

В результаті отримано розмір сформованого масиву векторів розміром (2021,1000), що свідчить про наявність 2021 рядку (опису), довжина кожного з яких становить 1000 символів. Для початкового навчання було обрано перші 1000 найчастіше вживаних слів в описах.

Для представлення тексту за допомогою методу TF–IDF було використано клас TfidfVectorizer бібліотеки sklearn.

Клас TfidfVectorizer з бібліотеки sklearn використовується для перетворення текстових даних в числову форму, щоб їх можна було використовувати для машинного навчання. TF–IDF (Term Frequency–Inverse Document Frequency) – це спосіб оцінки важливості термінів у тексті.

TF (частота терміну) вимірює, наскільки часто термін зустрічається у документі. IDF (обернена частота документа) вимірює, наскільки унікальним є термін у колекції документів. Чим рідше термін зустрічається у всій колекції, тим вищий його IDF. TfidfVectorizer об'єднує ці дві метрики, розраховуючи TF для кожного терміну у кожному документі та множачи його на IDF цього терміну по всій колекції документів. Результат – матриця, де кожен рядок представляє документ, а кожний стовпчик – термін, і кожна комірка – TF–IDF вага терміну у документі.

Клас TfidfVectorizer працює за наступним алгоритмом:

1) Токенезація. Відбувається розбиття тексту на окремі слова або токени. Наприклад, розбиття речення "Це дуже цікавий текст" на ["Це", "дуже", "цікавий", "текст"].

2) Побудова словнику. TfidfVectorizer будує словник всіх унікальних термінів у тексті, і кожен термін присвоює унікальний індекс.

3) Розрахунок показника TF. Для кожного документу визначається кількість входжень кожного терміну (слова) зі словника. Це може бути просто частота (кількість разів, як термін зустрічається в документі, яка розраховується за формулою (1.1)), або вагова форма, така як нормалізована частота

4) Розрахунок показника IDF. Після обчислення TF, обчислюється IDF за формулою (1.2), для кожного терміну у всьому корпусі документів. Це вимірює, наскільки часто термін зустрічається у всій колекції документів.

5) Обчислення TF–IDF. TF–IDF вага кожного терміну обчислюється як добуток значення TF та IDF для кожного терміну у кожному документі.

6) Нормалізація векторів. Вектори TF–IDF нормалізуються, щоб кожен документ мав одиничну довжину, що полегшує порівняння між ними.

Під час реалізації, в методі TfidfVectorizer також можна змінювати параметри, які повпливають безпосередньо впливають на формування словнику термінів:

max_features: Цей параметр визначає максимальну кількість унікальних термінів (слів), які враховуються у побудові словника. Використовується для обмеження розміру словника і пам'яті.

Параметр ngram_range вказує діапазон n–грам, які враховуються під час аналізу тексту. Наприклад, якщо ngram_range=(1, 2), то враховуються як окремі слова, так і пари слів (біграми).

Параметр tokenizer визначає функцію токенізації, яка розбиває текст на слова або токени. За замовчуванням використовується токенізація, заснована на пробілах.

Параметр max_df і min_df визначають верхню та нижню межі частоти термінів у документах. Наприклад, max_df=0.95 вказує, що термін не буде враховуватися, якщо він зустрічається в більш ніж 95% документів, а min_df=2 означає, що термін повинен зустрічатися принаймні в двох документах.

Параметр smooth_idf: Призначений для регуляризації IDF, щоб уникнути ділення на нуль. За замовчуванням включено.

Параметр sublinear_tf визначає, чи буде використовуватися логарифмічна шкала для обчислення TF. Це може допомогти зменшити вплив термінів, які часто зустрічаються.

Програмна реалізація представлення текстової інформації за допомогою класу TfidfVectorizer також схожа на реалізацію представлення за допомогою методу CountVectorizer:

```
tfidf_converter = TfidfVectorizer(max_features=1000)

X = tfidf_converter.fit_transform(df['title']).toarray()
```

Після здійснення представлення тексту, окрім розміру отриманого словника або слів, які до нього потрапили також можна отримати значення міри TF–IDF для певних слів. В табл. 2.6 наведено 10 слів, міра TF–IDF яких найбільша серед усіх інших слів для усього набору даних.

Таблиця 2.6

Вибірка 10 слів з сформованого словнику, які мають найвищу оцінку міри TF–IDF

Слово	Міра TF–IDF
Planet	0.032936
Episode	0.021075
Day	0.018690
Every	0.017578
Guy	0.017116
Short	0.017110
Try	0.015660
New	0.015109
College	0.014824
Life	0.014677

Сформований словник має аналогічний розмір в 1000 символів через використання аналогічних параметрів при векторизації, відповідно кожен документ з набору представлений вектором довжиною в 1000.

Для представлення тексту за допомогою методу Embeddings було використано клас KeyedVectorts.

Був завантажений набір векторів слів [16], який був сформований шляхом навчання моделі на великих корпусах даних. Розмір завантаженого документа

становить 400 тисяч слів, які представлені у вигляді числових векторів довжиною 50 кожний.

Для представлення текстової інформації була створена функція конвертації.

```
def text_to_vector(text):
    words = text.split()
    vectors = []
    for word in words:
        try:
            vectors.append(word_vectors[word])
        except KeyError:
            pass  # word not in vocabulary
    if vectors:
        return np.mean(vectors, axis=0)
    else:
        return np.zeros(word_vectors.vector_size)
```

Дана функція приймає текстову строку в якості вхідного параметру та конвертує її в числовий вектор, використовуючи масив векторів GloVe. Для кожного слова відбувається пошук в попередньо завантаженому масиві GloVe, якщо слово в словнику відсутнє – воно не розглядається. В кінці функція повертає середній вектор усіх слів, які були присутні в поданому документі (строці).

```
glove_file = 'glove.6B.50d.txt'

word_vectors = KeyedVectors.load_word2vec_format(glove_file)

df['description_vector'] = df['title'].apply(text_to_vector)

X = np.vstack(df['description_vector'].values)
```

Після створення функції, був завантажений готовий корпус даних, який записаний у відповідну змінну. Використовуючи попередньо описану функцію, до стовбця, який містить текстові описи відео, було отримано векторні представлення текстової інформації. На рис. 2.8 наведено довільний опис відео, представлений з використанням підходу GloVe.

```
[ 2.78141260e-01  4.23635036e-01 -1.63260669e-01 -1.38026197e-02
  1.80964991e-01  1.23973131e-01  1.02844983e-01  9.94942486e-02
  4.98462431e-02 -2.39207372e-01  2.06141114e-01  5.74093759e-01
 -2.44456515e-01  5.82972318e-02  1.36143893e-01 -1.35195374e-01
 -7.16779977e-02  4.43100855e-02 -1.34678507e+00  1.00247994e-01
 -3.77577484e-01  1.27010003e-01  2.43965358e-01 -1.75903738e-01
  2.53050387e-01 -1.14523005e+00 -1.50146246e-01  1.90012604e-02
  1.41242504e-01 -1.99567258e-01  1.39654374e+00 -2.43646279e-02
 -3.98453861e-01  1.83511734e-01 -6.37494959e-05  1.58902377e-01
  2.50362486e-01 -4.63520020e-01 -3.06657463e-01 -4.77519512e-01
  2.62994856e-01  2.27444954e-02  7.91762099e-02 -3.40551287e-01
  1.97572231e-01 -8.34820047e-02  2.68852115e-01 -8.74217451e-01
  1.78713739e-01 -3.95112485e-01]
```

Рисунок 2.8 Довільний опис, представлений за допомогою GloVe

Таким чином, було реалізовано програмно представлення тексту, застосовуючи 3 підходи: Bag–of–Words, TF–IDF та GloVe. Кожен метод реалізований мовою програмування Python за допомогою відповідних класів, представлених різними бібліотеками. При реалізації кожного з методів було використано параметри за замовчуванням, дослідження щодо отриманих результатів з зміною параметрів методів наведені в наступних розділах дипломної роботи.

2.3 Програмна реалізація генерації ключових слів відео за його описом

Перед переходом до навчання моделі, був розроблений інтерфейс для користувача. Причиною цьому є реалізована можливість обирати необхідний користувачу алгоритм машинного навчання – RandomForestClassifier, XGBoosting або LinearSVC. Для цього у користувача запитується назва

алгоритму, який він обрав, після чого в коді реалізується саме обраний алгоритм машинного навчання.

Генерація ключових слів відбувається з використанням трьох методів представлення: Bag–of–Words, TF–IDF та GloVe. Описи, що визначають кожну з груп ключових слів векторизуються окремим методом. Перша група слів класифікується з використанням підходу Bag–of–Words, друга група слів з використанням підходу TF–IDF, третя група слів з використанням підходу GloVe. Загальну точність згенерованих ключових слів можна буде розрахувати визначивши середню помилку трьох навчених моделей.

Наступним кроком, дані, які було відформатовано та представлено, було розбито на навчальну та тестову вибірку (навчальна становить 80%, тестова відповідно 20% від загального обсягу) для кожного з алгоритмів:

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

Після розбиття на навчальну та тестову вибірку, створюється модель відповідного алгоритму машинного навчання з початковими параметрами:

1) Для RandomForestClassifier:

classifier = RandomForestClassifier(n_estimators=200, random_state=0)

2) Для XGBClassifier:

classifier = xgb.XGBClassifier(n_estimators=200, learning_rate=0.5, random_state=0)

3) Для LinearSVC:

classifier = LinearSVC(random_state=0)

Перед поданням набору даних для навчання моделі з використанням алгоритму XGBClassifier, їх необхідно додатково закодувати з використанням класу LabelEncoder(). Після отримання прогнозованих ключових слів з використанням обраної моделі, прогнози потрібно буде повернути до початкового вигляду, використовуючи метод inverse_transform.

Після навчання моделей, було проведено оцінку їх якості, порівнюючи прогнозований моделлю результат за тестовою вибіркою з фактичними

значеннями тестової вибірки. Аналогічно розрахована точність за навчальною вибіркою.

y_pred_test = classifier.predict(X_test)

y_pred_train = classifier.predict(X_train)

В результаті отримано показники метрики точності для кожної моделі, які наведено в табл. 2.7. Для зручності сприйняття, дані було візуалізовано за допомогою гістограми, наведеної на рис. 2.9.

Таблиця 2.7

Оцінка якостей моделей з використанням метрики точності

Алгоритм машинного навчання	Метод представлення	Точність за тренувальною вибіркою	Точність за тестовою вибіркою
RandomForestClassifier	Count Vectorizer	97.68%	65.04%
	TF–IDF	98.16%	74.57%
	GloVe	99.89%	60.59%
XGBClassifier	CountVectorizer	86.04%	64.78%
	TF–IDF	86.38%	61.21%
	GloVe	99.89%	63.7%
LinearSVC	CountVectorizer	95.88%	66.32%
	TF–IDF	97.73%	75.01%
	GloVe	60.08%	58.67%

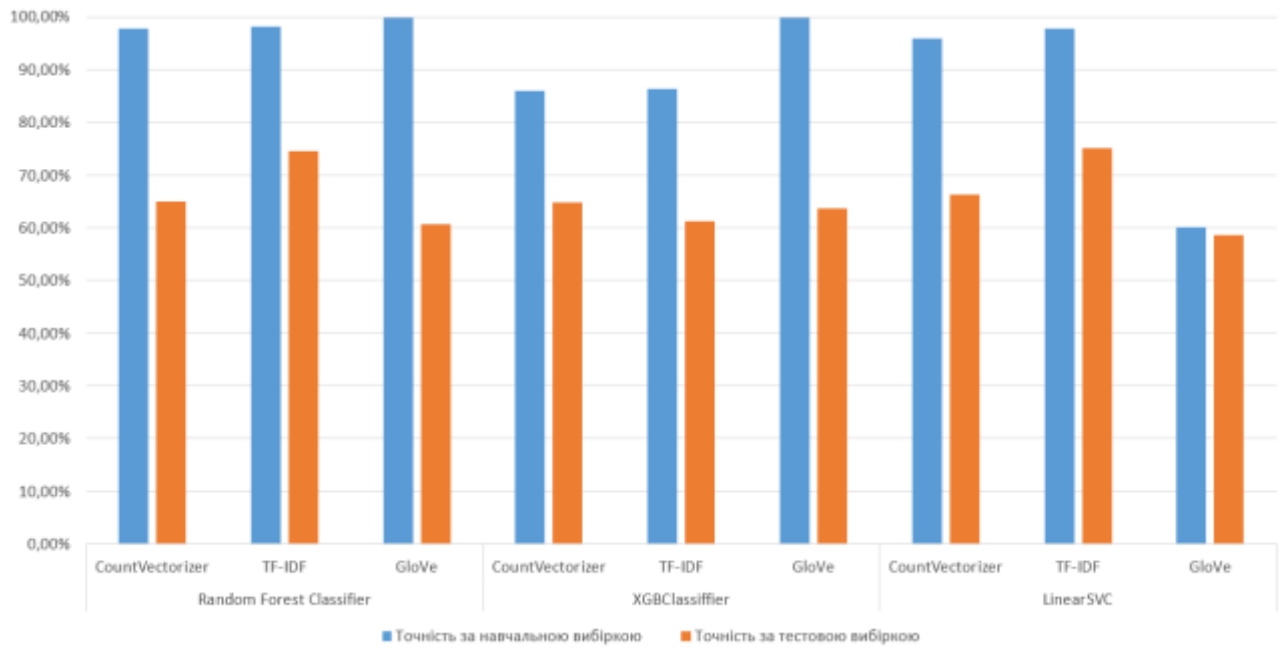

Рисунок 2.9 Значення параметру точності для навчених моделей

Отримані результати свідчать про те, що з використанням параметрів за замовчуванням в середньому отримано задовільну точність для всіх моделей. Можна простежити в середньому велику різницю між значеннями точності моделей на тренувальній та тестовій вибірці, що свідчить або про перенавченість моделей, або про недостатню кількість даних для навчання. Обидві проблеми можуть бути усунені шляхом змінення параметрів класів представлення та гіперпараметрів моделей.

Для додаткової перевірки було сформовано опис відео, який визначає що в відео описується розповідь про певні математичні розрахунки які допомагають при створенні кіно: «Science News. From Sine Waves to Cinematic Masterpieces: Studying Math in Film»

Його було відформатовано та подано до попередньо навчених моделей для отримання прогнозованих ключових слів. За допомогою методу predict, отримано відповідні ключові слова з кожної моделі, наведені в табл. 2.8.

Таблиця 2.8

Згенеровані ключові слова кожною за кожною моделлю для наданого довільного текстового опису відео

Алгоритм машинного навчання	RandomForestClassifier	XGBClassifier	LinearSVC
Слово №1	Technology	Science	Space
Слово №2	Math	Math	Education
Слово №3	News	News	News

Можна простежити, що усі три навчені моделі зуміли правильно передбачити тип поданого матеріалу – новини. Моделі з використанням випадкового лісу та градієнтного бустінгу правильно визначили напрямок – математика та основну тематику – технології та наука. Модель з використанням алгоритму LinearSVC правильно визначили лише одне ключове слово, але точність моделі не сильно відрізняється від інших використаних (див. табл. 2.7), що свідчить про додаткову необхідність підбору оптимальних гіперпараметрів при навчанні моделі.

Отже, після форматування даних та подачі їх до алгоритмів машинного навчання з початковими параметрами було здійснено навчання моделей. Використовуючи отримані моделі, було визначено їх точність на навчальній та тестовій виборці використовуючи метрику точності. В пункті 1.4 дипломної роботі визначено недоцільність використання метрики точності як остаточного засобу оцінки моделей багатокласової класифікації, що зумовлює необхідність визначення додаткових метрик якості моделей. Також було сформовано довільний опис відео, який подано для навчених моделей для генерації ключових слів. Отриманий результат є добрим, але може бути покращений шляхом зміни параметрів класів представлення тексту та гіперпараметрів моделей алгоритмів машинного навчання. Лістинг програмної реалізації представлення текстової інформації, а також застосування алгоритмів машинного навчання для вирішення задачі наведено в ДОДАТКУ Е.

2.4 Дослідження впливу зміни гіперпараметрів моделей при навчанні на їх якість

Для аналізу якості навчених моделей було використано матрицю помилок та метрики precision, recall.

Матриця помилок, або confussion matrix була побудована за допомогою метода confussion_matrix з використанням методу візуалізації hitmap для оцінки кожної з навчених моделей. Для зручності, числові показники були переведені у значення діапазону [0;1] (де 0 – це 0%, 1 – відповідно 100%). Побудована матриця помилок для моделі RandomForestClassifier з використанням підходу представлення bag–of–words наведена на рис. 2.10.

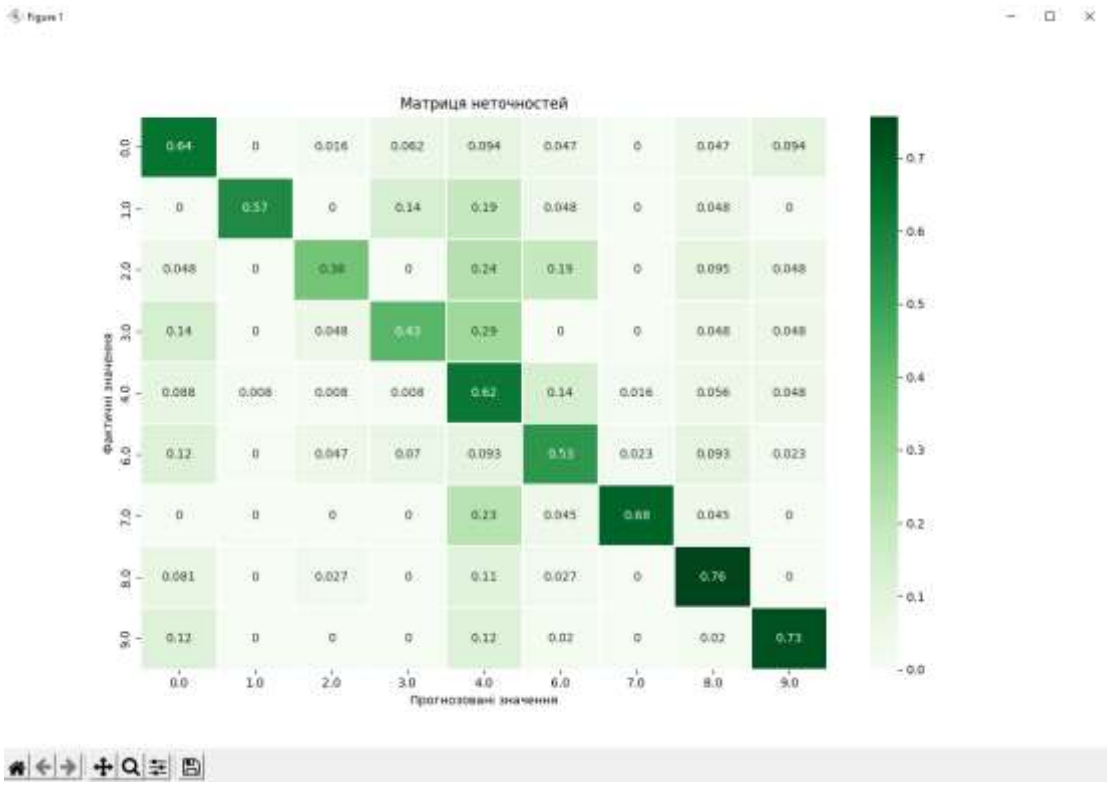

Рисунок 2.10 Матриця неточностей для моделі алгоритму RandomForestClassifier з використанням представлення Bag–of–Words

На представленому прикладі можна побачити будову матриці помилок для моделі багатокласової класифікації. Зліва розташовуються фактичні значення, знизу – значення які були прогнозовані. Відповідна на головній діагоналі сформованої матриці знаходиться вірогідність, з якою прогнозоване значення збігається з фактичним. Інші значення свідчать про помилкові прогнози, які не збігаються з фактичним значенням.

Можна визначити, що слова за номерами 1, 7, 8 та 9, відповідно discovery, software, pc, space найчастіше передбачаються правильно, а слова з номерами 2, 3, відповідно technology, engineering частіше передбачаються невірно. Це не пов’язано з кількісним розподілом класів (див. табл. 2.5), а отже проблема точності полягає у налаштування інструменту представлення та гіперпараметрах моделей.

Аналогічний розподіл вірогідностей для визначених класів простежується на матрицях неточностей для алгоритмів XGBClassifier та LinearSVC.

Після побудови матриці неточностей, було проведено аналіз метрик precision, recall та f–критерію за допомогою методу classification_report бібліотеки sklearn, які наведені в табл. 2.9.

Таблиця 2.9

Визначені метрики оцінки якості моделі що класифікує описи за першою групою слів за допомогою методу classification_report

Мітка	Precision	Recall	f–критерій
Science	0.59	0.64	0.61
Discovery	0.92	0.57	0.71
Technology	0.57	0.38	0.46
Engineering	0.45	0.43	0.44
Review	0.66	0.62	0.64
Invention	0.51	0.52	0.54
Tech	0.44	0.53	0.48
Software	0.83	0.68	0.75
PC	0.58	0.76	0.66
Space	0.71	0.73	0.72
Середнє значення	0.64	0.59	0.61

Отримані результати перекликаються з наведеними на матриці помилок (див. рис. 2.10). Слова technology та engineering мають найменші значення середнього гармонійного точності та повноти. Результати обчислення метрик дають змогу простежити, що для класу tech у моделі існує проблема в можливості відрізнити його від інших класів, про що свідчить низьке значення метрики precision, а для класу invention виникають складнощі при спробі виділення його серед інших, про що свідчить низьке значення recall.

Покращення результатів передбачення моделей було досягнуто шляхом зміни параметрів при представленні тексту, та за допомогою зміни гіперпараметрів моделей машинного навчання.

При розгляді підходу bag–of–words, з урахуванням розміру вибірки, яка становить 2000 строк, було визначено, що оптимальний розмір словника становить 500, отже параметр max_features було встановлено 500. Зміна параметрів max_df = 0.8, min_df = 5, дозволила враховувати слова, які зустрічаються менше ніж в 80% усіх розглянутих документах, при цьому не враховувати ті, що зустрічаються більш ніж в 5 документах. Це дозволило позбавитися слів, за якими моделі складно визначити до якого саме класу належить опис відео, що класифікується. Аналогічні параметри актуальні при використанні методу TF–IDF.

При застосуванні зазначених параметрів, точність навчених моделей, визначена на тестових вибірках зберіглася, а точність на навчальних – зменшилася. Це свідчить про зменшення впливу недостатньої кількості даних на кінцеву спроможність моделі узагальнювати. При розгляді отриманих оцінок якості моделі за нових налаштувань визначено збільшення метрик precision та recall. Це свідчить про те, що метрика точності моделі залишилася незмінною, отримана модель зможе ефективніше визначати окремі класи та відрізняти ці класи від інших.

При аналізі впливу зміни гіперпараметрів моделі на її якість, спершу був розглянутий алгоритм випадкового лісу. Для цього алгоритму змінювалися два параметри n_estimators, який визначає кількість дерев рішень в ансамблі та max_depth – глибина кожного дерева. Розглядаючи можливі значення кількості дерев в ансамблі, необхідно обрати таке значення, яке знаходиться між областю стану недонавчання та перенавчання. Для визначеного розміру навчальної вибірки в 2000 строк значення кількості дерев n_estimators 50 є замалим через зменшення точності, але при його поступовому збільшенні до 200 можна

простежити збільшення точності моделі. Збільшення кількості дерев в ансамблі більше 200 вже не впливає на точність, але може призвести до перенавченості моделі, тому ансамбль з 200 дерев буде вважатися оптимальним для узятої вибірки даних. На рис. 2.11 наведено залежність значення метрики точності моделі від кількості використаних дерев в ансамблі.

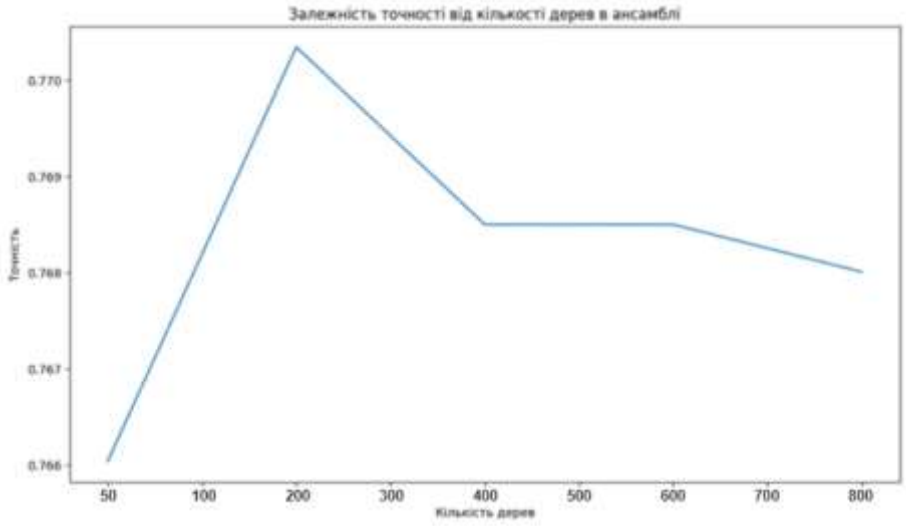

Рисунок 2.11 Графік залежності точності моделі від кількості дерев в ансамблі

З урахуванням розміру вибірки, та визначеної оптимальної кількості дерев рішень в ансамблі, наступним кроком було вирішено обмежити глибину будуємих дерев за допомогою параметру max_depth. Цей параметр дозволяє контролювати перенавченість моделі. За замовченням вузли моделі розширюються допоки не буде досягнутий максимальний розмір вибірки, що може призвести до хибних результатів за умови невеликого розміру вибірки.

Початково, глибина дерев була обмежена значенням 5, що призвело до зменшення точності моделі, яка становила 0,402, або 40,2%, що є близьким до вгадування. Наступним кроком максимально допустима глибина була збільшена до 25, в цьому випадку точність прогнозування збільшилася до 0,59% = 59%, цей показник також не є задовільним, але очевидно є кращим за попередній. Зміна точності зі збільшенням допустимої глибини дерев пов’язаний з розміром вибірки. Цей параметр був підібраний таким чином, аби

передбачити можливість перенавченості на малій виборці, але не обмежувати можливості моделі навчатися у разі подання до алгоритму більшого обсягу даних. Візуалізація залежності значення метрики точності на тестовій та навчальній вибірках наведена на рис. 2.12.

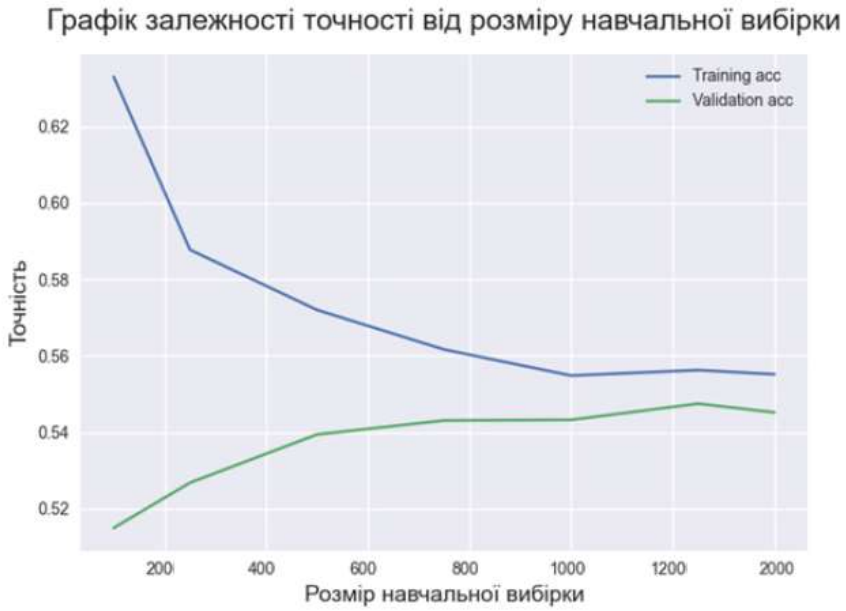

Рисунок 2.12 Графік залежності точності моделі від розміру навчальної вибірки (за умови що глибина дерев обмежена в 50)

В результаті було визначено, що оптимальним значенням глибини дерев є 50, при цьому різниця похибки за навчальною та тестовою вибірками є мінімальною, що свідчить про можливість моделі узагальнювати інформацію.

Точність моделей з визначеними параметрами та використовуючи кожен з вказаних методів представлення в середньому становить 0.708 = 70,8% що є гарним результатом для моделі класифікації.

Другим розглянутим алгоритмом машинного навчання був XGBClassifier. Як вже зазначалося, моделі, створені за допомогою алгоритму градієнтного бустінгу, є чутливими до зміни гіперпараметрів. В даному випадку розглядалося зміна параметрів learning_rate, який визначає крок, відповідно до якого корегуються ваги в кожній ітерації, max_depth, який дозволяє визначити, наскільки складні взаємозв’язки може визначити модель та gamma, що визначає

мінімальне зменшення функції втрат, необхідне для подальшого розбиття вузлів моделі.

Параметр learning_rate був визначений як найбільш впливовий на точність і був досліджений в першу чергу. Обиралися параметри в діапазоні від 0.01 до 1, в результаті аналізу результатів, оптимальним значенням цього параметру було визначено 0.1. На рис. 2.13 наведено залежність значень метрики точності моделі від значень параметру learning rate.

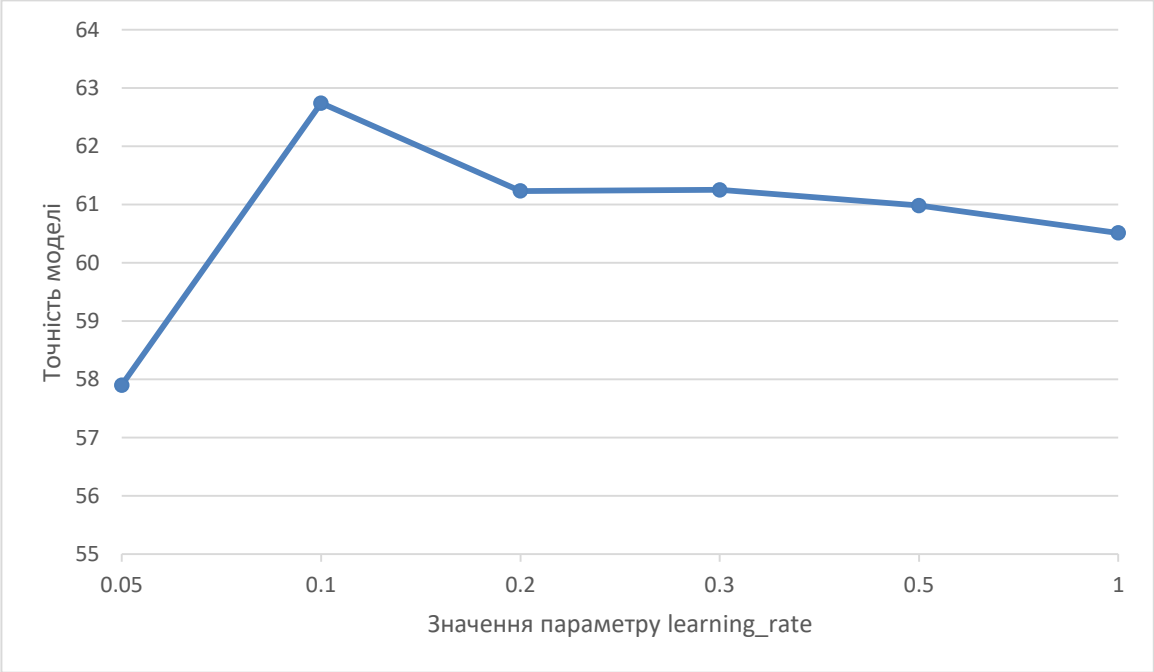

Рисунок 2.13 Графік залежності точності моделі від значення параметру learning_rate

При визначенні параметрів max_depth та gamma відповідно 10 та 0.1 вдалося збільшити точність моделі до середнього значення для кожного з методів представлення 0.682 = 68,2%.

Останнім розглянутим алгоритмом машинного навчання є LinearSVC. При аналізі можливих шляхів покращення результату з використанням цього алгоритму, було визначено необхідність зміни параметру C, який відповідає за регулювання жорсткості регуляризації, зменшення цього параметру підсилюють регуляризацію, більше – відповідно послаблюють. Регуляризація дозволяє підвищити здатність моделі узагальнювати інформацію на невеликих наборах даних та уникнути перенавченості на великих. Також досліджувалася

зміна функції втрат, яка визначається параметром loss. Параметр може приймати два значення – ‘hinge’, стандартна функція втрат для методу опорних векторів, або – ‘squared_hinge’ – варіація з квадратичними членами.

Експериментально було визначено, що оптимальним параметром жорсткості регуляризації є 0.15, а оптимальною функцією втрат – ‘squared_hinge’. З використанням цих параметрів, середній показник точності моделі за трьома методами представлення сягає 0.684 = 68,4%.

За результатами проведеного дослідження, сформовано оновлену таблицю (див. табл. 2.7) з показниками точності при використанні зазначених алгоритмів машинного навчання та методів представлення з урахуванням визначених оптимальних параметрів. Результаті оновлених значень наведені в табл. 2.10, візуалізовано за допомогою гістограми на рис. 2.14.

Таблиця 2.10

Оцінка якості моделей з використанням метрики точності, з урахуванням визначених оптимальних параметрів

Алгоритм машинного навчання	Метод представлення	Точність за тренувальною вибіркою	Точність за тестовою вибіркою
RandomForestClassifier	Count Vectorizer	76.67%	70.8%
	TF–IDF	75.4%	65.8%
	GloVe	99.89%	60.59%
XGBClassifier	CountVectorizer	77.72%	67.47%
	TF–IDF	82.98%	65.48%
	GloVe	99.89%	62.07%
Linear SVC	CountVectorizer	79.52%	68.4%
	TF–IDF	81.73%	71.89%
	GloVe	58.37%	59.99%

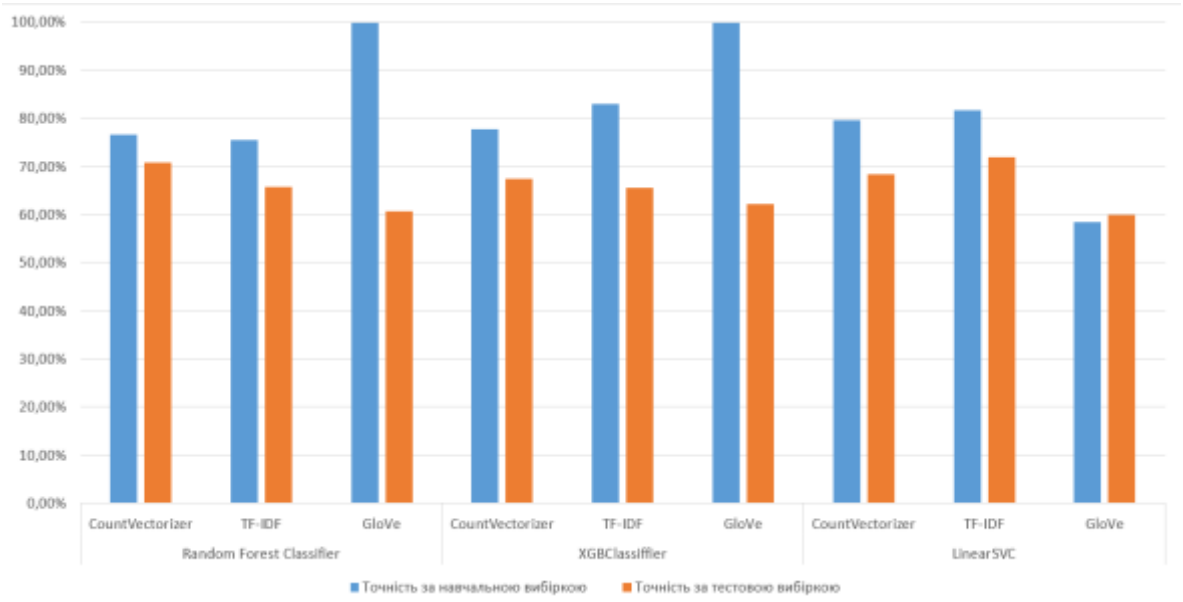

Рисунок 2.14 Значення параметру точності для моделей зі зміненими параметрами

В результаті проведеного дослідження, було визначено оптимальні комбінації алгоритму машинного навчання та методу представлення. Модель RandomForestClassifier в поєднанні з методом представлення bag–of–words, реалізованого за допомогою класу CountVectorizer має оцінку точності 70.8% що є гарним результатом для моделі класифікації. Результат точності моделі на навчальній вибірці становить 76.6%, невелика різниця між точністю на навчальній та тестовій вибірках свідчить про гарну здібність моделі до узагальнення та відсутність перенавченості. Аналогічно гарні результати отримані при використанні моделі LinearSVC та методу TF–IDF – 71.89%. Точність, отримана при використанні моделі XGBClassifier з методами bag–of–words та TF–IDF є гарною, ця модель відрізняється від інших швидкістю навчання та здійснення прогнозування. Метод представлення GloVe в комбінації з кожним алгоритмом надав задовільний результат точності, що зумовлюється недостатньою кількістю вибірки для отримання достатньо якісної моделі.

Таким чином, було покращено результати точності розробленого програмного додатку, визначено оптимальні комбінації підходів представлення тексту та алгоритмів машинного навчання. Отриманий програмний додаток має

гарну точність, та дає змогу згенерувати актуальні ключові слова за описом відео.

2.5 Висновки до розділу 2

В другому розділі дипломної роботи було здійснено аналіз вхідних даних з використанням мови програмування Python, за допомогою проведеного аналізу було визначено оптимальну стратегію здійснення генерації ключових слів відео за його описом. Стратегія полягає у здійсненні попереднього автоматичного анотування відео за трьома сформованими групами ключових слів. Це дозволило уникнути використання даних які є хибними та потенційно перешкоджали б навчанню моделей. Враховуючи мету, обрана стратегія надає можливість враховувати відео, які безпосередньо стосуються обраної теми.

Після визначення стратегії вирішення задачі, був реалізований програмний додаток який її вирішує. Для реалізації необхідних методів представлення та алгоритмів машинного навчання застосовано бібліотеки sklearn, xgb та genism. Вхідні дані було відформатовано, розділено на навчальну та тестову вибірку та подано до алгоритмів машинного навчання. Отримані моделі було оцінено розглянутими в розділі 1 метриками оцінки якостей моделей, висунуто відповідні припущення щодо можливостей їх покращити.

Був проведений аналіз впливу зміни параметрів методів представлення та алгоритмів машинного навчання на оцінки якостей моделей. Визначено оптимальні параметри які покращують прогнозовані результати. Була сформована таблиця з отриманою точністю для кожної комбінації алгоритм – метод представлення, за якої визначено які з цих комбінацій є кращими.

ВИСНОВОК

У дипломній роботі розв’язано актуальну прикладну задачу класифікації текстової інформації з застосуванням штучного інтелекту. Розроблено програмний додаток, який автоматично генерує ключові слова для відео наукової тематики на основі його текстового опису, використовуючи штучний інтелект. Це рішення забезпечує ефективну підтримку популяризації наукових відео на платформі YouTube, підвищуючи їхню видимість та доступність для цільової аудиторії.

Підбиваючи підсумки, вдалося досягти таких результатів:

1) Проведено аналіз актуальних шляхів вирішення задач класифікації текстової інформації з використанням штучного інтелекту. Визначено необхідні інструменти для вирішення задач класифікації.

2) Наведено ґрунтовний опис роботи визначених методів представлення та алгоритмів машинного навчання. Обґрунтовано вибір використаних технологій.

3) Здійснено аналіз визначених вхідних даних, на основі якого сформовано стратегію вирішення поставленої задачі.

4) Розроблено програмний додаток, який за допомогою описаних методів представлення та алгоритмів машинного навчання здатен генерувати ключові слова, які описують відео наукової тематики основуючись на описі цього відео.

5) Проведено порівняльний аналіз отриманих результатів, за допомогою якого було визначено оптимальні комбінації розглянутих методів для досягнення більшої точності згенерованих слів.

Результати даної роботи мають значну практичну користь. Розроблений додаток може бути використаний для автоматизації процесу оптимізації відео контенту на YouTube, що сприятиме більш ефективному поширенню наукових знань та залученню широкої аудиторії. Це, в свою чергу, підтримує розвиток науки та освіти в цифровому просторі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. S. J. The Interaction of Artificial Intelligence: How Terry Winograd Designed the SHRDLU Program. Medium. URL: https://medium.com/@staneyjoseph.in/the–interaction–of–artificial–intelligence–how–terry–winograd–designed–the–shrdlu–program–472a7de6d351 (Дата звернення 03.05.2024).

2. Rule–based word clustering for text classification / H. Han et al. the 26th annual international ACM SIGIR conference, Toronto, Canada, 28 July – 1 August 2003. New York, New York, USA, 2003. URL: https://doi.org/10.1145/860435.860543 (Дата звернення: 04.05.2024).

3. Medical prescription classification: a NLP–based approach / V. Carchiolo et al. 2019 Federated Conference on Computer Science and Information Systems, 1–4 September 2019. 2019. URL: https://doi.org/10.15439/2019f197 (Дата звернення: 04.05.2024).

4. ViTag: Automatic Video Tagging Using Segmentation and Conceptual Inference / A. A. Patwardhan et al. 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM), Singapore, Singapore, 11–13 September 2019. 2019. URL: https://doi.org/10.1109/bigmm.2019.00–12 (Дата звернення: 04.05.2024).

5. Kyröläinen A.–J., Laippala V. Predictive keywords: Using machine learning to explain document characteristics. Frontiers in Artificial Intelligence. 2023. Vol. 5. URL: https://doi.org/10.3389/frai.2022.975729 (Дата звернення: 09.05.2024).

6. Rouse M. What is Bag of Words and how is this strategy used in machine learning?. LinkedIn: Log In or Sign Up. URL: https://www.linkedin.com/pulse/what–bag–words–how–strategy–used–machine–learning–margaret–rouse (Дата звернення: 09.04.2024).

7. Qader W. A., Ameen M. M., Ahmed B. I. An Overview of Bag of Words; Importance, Implementation, Applications, and Challenges. 2019 International Engineering Conference (IEC), Erbil, Iraq, 23–25 June 2019. 2019. URL: https://doi.org/10.1109/iec47844.2019.8950616 (Дата звернення: 09.05.2024).

8. Luhn H. P. The Automatic Creation of Literature Abstracts. IBM Journal of Research and Development. 1958. Vol. 2, no. 2. P. 159–165. URL: https://doi.org/10.1147/rd.22.0159 (Дата звернення: 09.05.2024).

9. Bijoyan Das, Sarit Chakraborty. An Improved Text Sentiment Classification Model Using TF–IDF and Next Word Negation. 2018. 6p.

10. Pavlov Y. L. Random forests. De Gruyter, Inc., 2019. 122 p.

11. D. E. Johnson, F. J. Oles, T. Zhang, T. Goetz, “A decision–tree–based symbolic rule induction system for text categorization”, IBM Systems Journal, September 2002.

12. Shanahan J. and Roma N., Improving SVM Text Classification Performance through Threshold Adjustment, LNAI 2837, 2003, 361–372.

13. Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P., “SMOTE: Synthetic Minority Over–sampling Technique,” Journal of AI Research, 16 2002, pp. 321–357.

14. YouTube Trending Video Dataset (updated daily). Kaggle: Your Machine Learning and Data Science Community. URL: https://www.kaggle.com/ datasets/rsrishav/youtube–trending–video–dataset (Дата звернення: 02.05.2024).

15. Yemets M.O., Khabarlak K.S. Video keyword generation from text description based on artificial intelligence // I (VII) міжнародна науково-практична конференція здобувачів вищої освіти і молодих учених «Інформаційні

технології: теорія і практика» (20 – 22 березня 2024 р). Збірник тез. Дніпро. НТУ «ДП». – 168-170 с.

16. GloVe: Global Vectors for Word Representation. The Stanford Natural Language Processing Group. URL: https://nlp.stanford.edu/projects/glove/ (Дата звернення: 27.05.2024).

Додаток А.

Відомість матеріалів кваліфікаційної роботи

№ з/п	Позначення	Найменування	Кількість аркушів	Примітки
1				
2		Документація		
3				
4	САУ.КР.YY.ZZ.ПЗ	Пояснювальна записка	N1	Формат А4
5				
6		Демонстраційний матеріал	N2	Презентація на CD–R
7				
8		Копія роботи	1	Диск CD–R
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

					САУ.КР.YY.ZZ.ДА.ПЗ.			
Змін.	Аркуш	№ докум.	Підпис	Дата				
Розроб.		Ємець М.О.			**Матеріали кваліфікаційної роботи**	Літ.	Аркуш	Аркушів
К. розд.		Хабарлак К.С.						
Керівн.		Хабарлак К.С.				НТУ «ДП», 12; 124–20–1		
Н.контр.		Хом'як Т.В.						
Зав. каф.		Желдак Т.А.						

Запис **САУ.КР.YY.ZZ.ПЗ** означає наступне:
САУ – код випускаючої кафедри;
КР – кваліфікаційна робота;
N1 – загальна кількість сторінок пояснювальної записки кваліфікаційної роботи з додатками;
N2 – кількість аркушів демонстраційного матеріалу (слайдів презентації);
YY – рік захисту кваліфікаційної роботи в ЕК (наприклад "22");
ZZ – номер теми студента в наказі про затвердження теми кваліфікаційної роботи (наприклад "06");
ПЗ – пояснювальна записка;

ДА – додаток А;
12 – код галузі «Інформаційні технології».

Додаток Б.

Відгук

на кваліфікаційну роботу бакалавра

студента групи *124 – 20 – 1*
спеціальності *124 Системний аналіз*

Тема кваліфікаційної роботи: Генерація ключових слів відео за текстовим описом на основі штучного інтелекту
Обсяг кваліфікаційної роботи _____ стор.
Мета кваліфікаційної роботи: _____
_____
Актуальність теми _____
_____
_____
Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності бакалавра спеціальності 124 Системний аналіз, оскільки _____
_____
_____
Виконані в кваліфікаційній роботі завдання відповідають вимогам ступеня бакалавра. Оригінальність наукових рішень полягає в _____
_____
_____
_____
Практичне значення результатів кваліфікаційної роботи полягає в _____
_____
_____
Висновки підтверджують можливість використання результатів роботи в _____
_____
_____
Оформлення пояснювальної записки та демонстраційного матеріалу до неї виконано згідно з вимогами. Роботу виконано самостійно, відповідно до завдання та у повному обсязі (*в разі невідповідності – вказати*)
У роботі відзначено такі недоліки: _____
_____
_____
Кваліфікаційна робота в цілому заслуговує оцінки:_____

З урахуванням висловлених зауважень автор (не) заслуговує присвоєння освітньої кваліфікації «бакалавр з системного аналізу».

Керівник кваліфікаційної роботи бакалавра,
науковий ступінь, вчене звання, посада _____ / ПІБ

Додаток В.

Рецензія

на кваліфікаційну роботу бакалавра

студента групи *124 – 20 – 1*
спеціальності *124 Системний аналіз*

Тема кваліфікаційної роботи: Генерація ключових слів відео за текстовим описом на основі штучного інтелекту
Обсяг кваліфікаційної роботи: _____
Висновок про відповідність кваліфікаційної роботи завданню та освітньо–професійній програмі спеціальності _____
_____
_____
Загальна характеристика кваліфікаційної роботи, ступінь використання нормативно–методичної літератури та передового досвіду
_____
_____
_____
Позитивні сторони кваліфікаційної роботи:
_____
_____
_____

Основні недоліки кваліфікаційної роботи:
_____
_____
_____
_____
Кваліфікаційна робота в цілому заслуговує оцінки: _____

З урахуванням висловлених зауважень автор (не) заслуговує присвоєння освітньої кваліфікації «бакалавр з системного аналізу».

Рецензент,
науковий ступінь, вчене звання, посада _____ / ПІБ

Додаток Г.

Лістинг програмного коду мовою Python, за допомогою якого було здійснено аналіз вхідних даних.

```
import matplotlib
import pandas as pd
import numpy as np
import matplotlib as mpl
from matplotlib import pyplot as plt
import seaborn as sns

import warnings
from collections import Counter
import datetime
import wordcloud
import json

from DefLibs import *

PLOT_COLORS = ["#268bd2", "#0052CC", "#FF5722", "#b58900", "#003f5c"]
pd.options.display.float_format = '{:.2f}'.format
sns.set(style="ticks")
plt.rc('figure', figsize=(8, 5), dpi=100)
plt.rc('axes', labelpad=20, facecolor="#ffffff", linewidth=0.4, grid=True, labelsize=14)
plt.rc('patch', linewidth=0)
plt.rc('xtick.major', width=0.2)
plt.rc('ytick.major', width=0.2)
plt.rc('grid', color='#9E9E9E', linewidth=0.4)
plt.rc('font', family='Arial', weight='400', size=10)
plt.rc('text', color='#282828')
plt.rc('savefig', pad_inches=0.3, dpi=300)

df = pd.read_csv('US_youtube_trending_data.csv')  # Завантажуємо дані
print(df.info())  # Отримуємо загальну інформацію про датасет

# Позбавляємося від рядків з нал
df["description"] = df["description"].fillna(value="")

# Дослідимо датасет по роках
cdf = df["trending_date"].apply(lambda x: '20' + x[:2]).value_counts().to_frame().reset_index().rename(
    columns={"trending_date": "Рік", "count": "Кількість"})

print(cdf.head())

print(df.describe())


# 1. Кількість даних за категоріями
category_counts = df['categoryId'].value_counts().to_frame().reset_index().rename(columns={'categoryId': "Категорія", "count":
"Кількість"})
print(category_counts.head())

read_txt_file("youtube_api_video_category_id_list.txt")
categories_dict = read_txt_file(file_path)

category_counts['Категорія'] = category_counts['Категорія'].replace(categories_dict)

plt.figure(figsize=(12, 8))

plt.pie(category_counts['Кількість'],
        labels=None,
        autopct='%1.1f%%',
        pctdistance=1.1)
plt.title('Распределение категорий')
plt.axis('off')
plt.legend(category_counts['Категорія'], loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```

```
# 2. Діаграма топ тегів за категоріями та глобально
tags_split = []
tags_row = []
tags_count = {}

#Глобально
tags_raw_counts = df['tags'].value_counts().to_frame().reset_index()

for index, row in tags_raw_counts.iterrows():
    tags_row = row['tags'].split('|')
    for tag in tags_row:
        if tag.lower() in tags_count:
            tags_count[tag.lower()] += row['count']
        else:
            tags_count[tag.lower()] = row['count']
tags_count.pop('[none]')
tags_count = dict(sorted(tags_count.items(), key=lambda x: x[1], reverse=True))
with open('tags_stats.txt', 'w', encoding="utf–8") as f:
    for tag, count in tags_count.items():
        f.write(f"{tag}: {count}\n")
        print(f"{tag}: {count}\n")

keys = list(tags_count.keys())[0:10]
values = list(tags_count.values())[0:10]

plt.bar(keys, values, color="red")
plt.title("Top tags chart (global)")
plt.bar(keys, values)
plt.xlabel("Tag")
plt.ylabel("Count")
plt.show()
#Entertaiment
tags_split = []
tags_row = []
tags_count = {}
dfEnt = df[df['categoryId'] == 24]
tags_raw_counts = dfEnt['tags'].value_counts().to_frame().reset_index()

for index, row in tags_raw_counts.iterrows():
    tags_row = row['tags'].split('|')
    for tag in tags_row:
        if tag.lower() in tags_count:
            tags_count[tag.lower()] += row['count']
        else:
            tags_count[tag.lower()] = row['count']
tags_count.pop('[none]')
tags_count = dict(sorted(tags_count.items(), key=lambda x: x[1], reverse=True))

keys = list(tags_count.keys())[0:10]
values = list(tags_count.values())[0:10]

plt.bar(keys, values, color="red")
plt.title("Діаграма топ тегів за категорією Entertaiment")
plt.bar(keys, values)
plt.xlabel("Тег")
plt.ylabel("Кількість")
plt.show()
#Gaming
tags_split = []
tags_row = []
tags_count = {}
dfEnt = df[df['categoryId'] == 20]
tags_raw_counts = dfEnt['tags'].value_counts().to_frame().reset_index()

for index, row in tags_raw_counts.iterrows():
    tags_row = row['tags'].split('|')
    for tag in tags_row:
```

```
        if tag.lower() in tags_count:
            tags_count[tag.lower()] += row['count']
        else:
            tags_count[tag.lower()] = row['count']
tags_count.pop('[none]')
tags_count = dict(sorted(tags_count.items(), key=lambda x: x[1], reverse=True))

keys = list(tags_count.keys())[0:10]
values = list(tags_count.values())[0:10]

plt.bar(keys, values, color="red")
plt.title("Діаграма топ тегів за категорією Gaming")
plt.bar(keys, values)
plt.xlabel("Тег")
plt.ylabel("Кількість")
plt.show()
#Music
tags_split = []
tags_row = []
tags_count = {}
dfEnt = df[df['categoryId'] == 10]
tags_raw_counts = dfEnt['tags'].value_counts().to_frame().reset_index()

for index, row in tags_raw_counts.iterrows():
    tags_row = row['tags'].split('|')
    for tag in tags_row:
        if tag.lower() in tags_count:
            tags_count[tag.lower()] += row['count']
        else:
            tags_count[tag.lower()] = row['count']
tags_count.pop('[none]')
tags_count = dict(sorted(tags_count.items(), key=lambda x: x[1], reverse=True))

keys = list(tags_count.keys())[0:10]
values = list(tags_count.values())[0:10]

plt.bar(keys, values, color="red")
plt.title("Діаграма топ тегів за категорією Music")
plt.bar(keys, values)
plt.xlabel("Тег")
plt.ylabel("Кількість")
plt.show()
# Sports
tags_split = []
tags_row = []
tags_count = {}
dfEnt = df[df['categoryId'] == 17]
tags_raw_counts = dfEnt['tags'].value_counts().to_frame().reset_index()

for index, row in tags_raw_counts.iterrows():
    tags_row = row['tags'].split('|')
    for tag in tags_row:
        if tag.lower() in tags_count:
            tags_count[tag.lower()] += row['count']
        else:
            tags_count[tag.lower()] = row['count']
tags_count.pop('[none]')
tags_count = dict(sorted(tags_count.items(), key=lambda x: x[1], reverse=True))

keys = list(tags_count.keys())[0:10]
values = list(tags_count.values())[0:10]

plt.bar(keys, values, color="red")
plt.title("Діаграма топ тегів за категорією Sports")
plt.bar(keys, values)
plt.xlabel("Тег")
plt.ylabel("Кількість")
plt.show()
```

```
# 3. Зміна топ тегів від часу
df = pd.read_csv('US_youtube_trending_data.csv')

df['publishedAt'] = pd.to_datetime(df['publishedAt'])


def count_tags(tags_string, tags_to_count):
    tag_counts = {tag: 0 for tag in tags_to_count}
    tags = tags_string.lower().split('|')
    for tag in tags:
        if tag in tags_to_count:
            tag_counts[tag] += 1
    return tag_counts


interesting_tags = ['challenge', 'vlog', 'minecraft', 'football']
df['tag_counts'] = df['tags'].apply(lambda x: count_tags(x, interesting_tags))

for tag in interesting_tags:
    df[tag] = df['tag_counts'].apply(lambda x: x[tag])

grouped_data = df.groupby([df['publishedAt'].dt.year, df['publishedAt'].dt.month])[interesting_tags].sum()

grouped_data.plot(kind='line', figsize=(10, 6))
plt.xlabel('Дата')
plt.ylabel('Кількість зустрічей')
plt.title('Зміна популярності тегів з часом')
plt.legend(title='Tags')
plt.grid(True)
plt.show()
```

Додаток Д.

Лістинг програмного коду мовою Python, за допомогою якого було вдосконалено початковий набір даним, шляхом здійснення автоматичної анотації.

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import ensemble
from sklearn.model_selection import learning_curve
import ITextFormatter as tf

first_group_keywords = {
    'science': 0,
    'discovery': 1,
    'technology': 2,
    'engineering': 3,
    'review': 4,
    'invention': 5,
    'tech': 6,
    'software': 7,
    'pc': 8,
    'space': 9
}
second_group_keywords = {
    'education': 0,
    'politics': 1,
    'geography': 2,
    'law': 3,
    'tutorial': 4,
    'math': 5,
    'economics': 6,
    'physics': 7,
    'social': 8,
    'investing': 9
}
third_group_keywords = {
    'history': 0,
    'art': 1,
    'documentary': 2,
    'mystery': 3,
    'film': 4,
    'animation': 5,
    'cartoon': 6,
    'news': 7,
    'stories': 8,
    'design': 9
}


def create_csv_with_key_words():
    def assign_first_group_number(tags):
        tag_list = tags.split('|')
        for tag in tag_list:
            if tag in first_group_keywords:
                return first_group_keywords[tag]
        return None

    def assign_second_group_number(tags):
        tag_list = tags.split('|')
        for tag in tag_list:
            if tag in second_group_keywords:
                return second_group_keywords[tag]
        return None

    def assign_third_group_number(tags):
        tag_list = tags.split('|')
        for tag in tag_list:
```

```
            if tag in third_group_keywords:
                return third_group_keywords[tag]
        return None

    us_df = pd.read_csv('US_youtube_trending_data.csv')
    gb_df = pd.read_csv('GB_youtube_trending_data.csv')

    df = pd.concat([us_df, gb_df])
    df.reset_index(drop=True, inplace=True)

    df = df[df['tags'] != '[None]']
    df = df.dropna(subset=['title'])
    df = df.drop_duplicates(subset='video_id')

    df['tags'] = df['tags'].str.lower()

    df['title'] = df['title'].apply(tf.remove_stopwords)
    df['title'] = df['title'].apply(tf.text_format)

    df['I_key_word'] = df['tags'].apply(assign_first_group_number)
    df['II_key_word'] = df['tags'].apply(assign_second_group_number)
    df['III_key_word'] = df['tags'].apply(assign_third_group_number)

    df_first_group = df.dropna(subset=['I_key_word']).copy()
    df_second_group = df.dropna(subset=['II_key_word']).copy()
    df_third_group = df.dropna(subset=['III_key_word']).copy()

    keywords_counts = df['I_key_word'].value_counts()
    print(keywords_counts)
    keywords_counts = df['II_key_word'].value_counts()
    print(keywords_counts)
    keywords_counts = df['III_key_word'].value_counts()
    print(keywords_counts)

    df_first_group.to_csv('first_US_data_with_keywords.csv', index=False)
    df_second_group.to_csv('second_US_data_with_keywords.csv', index=False)
    df_third_group.to_csv('third_US_data_with_keywords.csv', index=False)

    df.to_csv('all_US_data_with_keywords.csv', index=False)
    return df


def decode_keyword(group_number, number):
    if group_number == 1:
        keywords = first_group_keywords
    elif group_number == 2:
        keywords = second_group_keywords
    elif group_number == 3:
        keywords = third_group_keywords
    else:
        return "Invalid group number"

    for keyword, code in keywords.items():
        if code == number:
            return keyword

    return "Keyword not found"
```

Додаток Е.

Лістинг програмного коду мовою Python, який реалізує генерацію ключових слів відео за його описом з використанням штучного інтелекту.

```
from collections import OrderedDict

import libs
import numpy as np
import pandas as pd
import os
import ITextFormatter as tf
import seaborn as sns
import xgboost as xgb

from matplotlib import pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.model_selection import train_test_split, learning_curve, cross_val_score, GridSearchCV
from sklearn.svm import LinearSVC
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models import KeyedVectors

N_max_features = 1000


# Зчитування та запис словника категорій
category_map = {}
with open('youtube_api_video_category_id_list.txt', 'r') as file:
    for line in file:
        category_id, category_name = line.strip().split(' ')
        category_map[int(category_id)] = category_name


def vector_to_text(cv, vector, N):
    inversed_text = []
    for row in vector[:N]:
        words = []
        for idx, count in enumerate(row):
            if count > 0:
                word = cv.get_feature_names_out()[idx]
                words.extend([word] * count)
        inversed_text.append(" ".join(words))
    return inversed_text


# Отримання прогнозованної категорії
def get_one_pred(y_pred):
    return [category_map.get(pred, 'Unknown') for pred in y_pred]


# Отримання трьох можливих категорій
def get_three_proba_pred(classifier, vectorized_test):
    probabilities = classifier.predict_proba(vectorized_test)
    top_categories_indices = np.argsort(probabilities, axis=1)[:, –3:]
    top_categories = [[classifier.classes_[i] for i in indices] for indices in top_categories_indices]
    predicted_categories = []
    for pred_list in top_categories:
        category_list = []
        for pred in pred_list:
            category = category_map.get(pred, 'Unknown')
            category_list.append(category)
        predicted_categories.append(category_list)
    return predicted_categories
```

```
def build_confussion_matrix(y_pred, y_test):
    labels = np.unique(y_pred)
    matrix = confusion_matrix(y_test, y_pred)
    matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]
    plt.figure(figsize=(10, 8))
    sns.heatmap(matrix, annot=True, annot_kws={'size': 10},
                cmap=plt.cm.Greens, linewidths=0.2, xticklabels=labels, yticklabels=labels)
    plt.xlabel('Прогнозовані значення')
    plt.ylabel('Фактичні значення')
    plt.title('Матриця неточностей')
    plt.show()

if os.path.isfile('all_US_data_with_keywords.csv'):
    print("FILE EXIST.")

else:
    print("FILE DOESNT EXIST. CREATING FILE...")
    df = libs.create_csv_with_key_words()

choice = input("Оберіть необхідний алгоритм машинного навчання (rf, gb or lsvc): ")

print("———————————————————————————— CountVectorizer ————————————————————————————")
#
df = pd.read_csv('first_US_data_with_keywords.csv')

cv_descr = CountVectorizer(max_features=N_max_features, min_df=5, max_df=0.8)
res_cv = cv_descr.fit_transform(df['title'])

# Вивід отриманого словнику
print(res_cv.shape)  # Розмір матриці термінів
print(cv_descr.vocabulary_)  # Сформований словник (слово : його номер на векторі)

X = res_cv.toarray()  # Приведення словника до масиву
Y = df['I_key_word'].tolist()
le = LabelEncoder()

# Розподіл загальної вибірки на тестову та навчальну
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

if choice == "rf":
    classifier = RandomForestClassifier(n_estimators=200, max_depth=50, random_state=0)
    classifier.fit(X_train, y_train)
elif choice == "gb":
    # Gradient Boosting
    classifier = xgb.XGBClassifier(n_estimators=200, learning_rate=0.1, max_depth=10, gamma=0.1, random_state=0)
    classifier.fit(X_train, y_train)
elif choice == "lsvc":
    # LinearSVC
    classifier = LinearSVC(random_state=0, C=0.15, loss='squared_hinge')
    classifier.fit(X_train, y_train)
else:
    print("!!! WRONG ALGO NAME !!! (creating RF model)")
    classifier = RandomForestClassifier(n_estimators=200, random_state=0)
    classifier.fit(X_train, y_train)


# Виконання прогнозу за текстовую вибіркою
y_pred_test = classifier.predict(X_test)
y_pred_train = classifier.predict(X_train)

# Обчислення точності прогнозу
print("Точність моделі + (CountVectorizer) на тренувальній виборці (accuracy_score): " + str(
    round(accuracy_score(y_train, y_pred_train) * 100, 2)) + " %")
print("Точність моделі + (CountVectorizer) на тестовій виборці (accuracy_score): " + str(
    round(accuracy_score(y_test, y_pred_test) * 100, 2)) + " %")
# Побудова матриці помилок та її візуалізація за тестовою вибіркою
build_confussion_matrix(y_pred_test, y_test)
```

```
print(classification_report(y_test, y_pred_test))

# Виконання прогнозу на тестовому описі
test_text = "From Sine Waves to Cinematic Masterpieces: Exploring Math in Film"
test_formatted = tf.text_format(tf.remove_stopwords(test_text))
vectorized_test = cv_descr.transform([test_formatted]).toarray()
y_pred_example = classifier.predict(vectorized_test)

print("Спрогнозоване ключове слово за описом: ", libs.decode_keyword(1, y_pred_example))

print("———————————————————————————— TF – IDF ————————————————————————————————————")

df = pd.read_csv('second_US_data_with_keywords.csv')

tfidf_converter = TfidfVectorizer(max_features=500)
X = tfidf_converter.fit_transform(df['title']).toarray()
Y = df['II_key_word'].tolist()

# Получаем список всех слов
words = tfidf_converter.get_feature_names_out()

# Вычисляем средние значения TF–IDF для каждого слова
mean_tfidf = np.mean(X, axis=0)

# Получаем индексы слов с наибольшими средними значениями TF–IDF
top_indices = np.argsort(mean_tfidf)[::–1][:10]

# Выводим топ–10 слов с соответствующими оценками TF–IDF
print("Топ–10 слів з найбільшими оцінками TF–IDF:")
for idx in top_indices:
    word = words[idx]
    tfidf_score = mean_tfidf[idx]
    print(f"{word}: {tfidf_score}")

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
le = LabelEncoder()


if choice == "rf":
    classifier = RandomForestClassifier(n_estimators=200, max_depth=50, random_state=0)
    classifier.fit(X_train, y_train)
elif choice == "gb":
    classifier = xgb.XGBClassifier(n_estimators=200, learning_rate=0.1, max_depth=10, gamma=0.1, random_state=0)
    classifier.fit(X_train, y_train)
elif choice == "lsvc":
    # LinearSVC
    classifier = LinearSVC(random_state=0, C=0.15, loss='squared_hinge')
    classifier.fit(X_train, y_train)
else:
    print("!!! WRONG ALGO NAME !!! (creating RF model)")
    classifier = RandomForestClassifier(n_estimators=200, random_state=0)
    classifier.fit(X_train, y_train)

y_pred_test = classifier.predict(X_test)
y_pred_train = classifier.predict(X_train)

print("Точність моделі + (TF–IFD) на тестовій виборці (accuracy_score): " + str(
    round(accuracy_score(y_train, y_pred_train) * 100, 2)) + " %")
print("Точність моделі + (TF–IFD) на тестовій виборці (accuracy_score): " + str(
    round(accuracy_score(y_test, y_pred_test) * 100, 2)) + " %")

print(classification_report(y_test, y_pred_test))

# Побудова матриці помилок та її візуалізація
build_confussion_matrix(y_pred_test, y_test)

# Виконання прогнозу на тестовому описі
test_text = "From Sine Waves to Cinematic Masterpieces: Exploring Math in Film"
test_formatted = tf.text_format(tf.remove_stopwords(test_text))
```

```
vectorized_test = tfidf_converter.transform([test_formatted]).toarray()
y_pred_example = classifier.predict(vectorized_test)

print("Спрогнозоване ключове слово за описом: ", libs.decode_keyword(2, y_pred_example))


print("———————————————————————————————— GloVe ————————————————————————————————")

df = pd.read_csv('third_US_data_with_keywords.csv')

glove_file = 'glove.6B.50d.txt'

word_vectors = KeyedVectors.load_word2vec_format(glove_file)

# Трансформація тексту в числові вектори з використанням GloVe
def text_to_vector(text):
    words = text.split()
    vectors = []
    for word in words:
        try:
            vectors.append(word_vectors[word])
        except KeyError:
            pass  # word not in vocabulary
    if vectors:
        return np.mean(vectors, axis=0)
    else:
        return np.zeros(word_vectors.vector_size)

df['description_vector'] = df['title'].apply(text_to_vector)

X = np.vstack(df['description_vector'].values)

print(df['description_vector'].iloc[0])

Y = df['III_key_word'].tolist()

le = LabelEncoder()

# Розподіл загальної вибірки на тестову та навчальну
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)

if choice == "rf":
    classifier = RandomForestClassifier(n_estimators=200, max_depth=50, random_state=0)
    classifier.fit(X_train, y_train)
elif choice == "gb":
    classifier = xgb.XGBClassifier(n_estimators=200, learning_rate=0.1, max_depth=10, gamma=0.1, random_state=0)
    classifier.fit(X_train, y_train)
elif choice == "lsvc":
    # LinearSVC
    classifier = LinearSVC(random_state=0, C=0.15, loss='squared_hinge')
    classifier.fit(X_train, y_train)
else:
    print("!!! WRONG ALGO NAME !!! (creating RF model)")
    classifier = RandomForestClassifier(n_estimators=200, random_state=0)
    classifier.fit(X_train, y_train)


# Виконання прогнозу за текстовую вибіркою
y_pred_test = classifier.predict(X_test)
y_pred_train = classifier.predict(X_train)

# Обчислення точності прогнозу
print("Точність моделі + (GloVe) на тренувальній виборці (accuracy_score): " + str(
    round(accuracy_score(y_train, y_pred_train) * 100, 2)) + " %")
print("Точність моделі + (GloVe) на тестовій виборці (accuracy_score): " + str(
    round(accuracy_score(y_test, y_pred_test) * 100, 2)) + " %")

print(classification_report(y_test, y_pred_test))
```

```
# Побудова матриці помилок та її візуалізація
build_confussion_matrix(y_pred_test, y_test)

# Виконання прогнозу на тестовому описі
test_text = "From Sine Waves to Cinematic Masterpieces: Exploring Math in Film"
test_formatted = tf.text_format(tf.remove_stopwords(test_text))
y_pred_example = classifier.predict(text_to_vector(test_text).reshape(1, –1))

print("Спрогнозоване ключове слово за описом: ", libs.decode_keyword(3, y_pred_example))
```