МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
_____
(підпис)

_____

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна технологія аналізу силабусів навчальних дисциплін із застосуванням великих мовних моделей»
здобувача групи ІН.м-23 Карпцова Андрія Сергійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Андрій КАРПЦОВ
_____
(підпис)

Керівник,
старший викладач кафедри
комп'ютерних наук, к.т.н
Борис КУЗІКОВ
_____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН.м-23 Карпцова Андрія Сергійовича

1. Тема роботи: «Інформаційна технологія аналізу силабусів навчальних дисциплін із застосуванням великих мовних моделей»
затверджую наказом по СумДУ від «20» листопада 2023 р. № 1308-VI

2. Термін здачі здобувачем кваліфікаційної роботи до 18 грудня 2023 року

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Проаналізувати способи видобутку текстової інформації з pdf файлів. 2) Провести аналіз наявних у відкритому доступі великих мовних моделей, їх API та можливість застосування для задачі видобутку даних із силабусів дисциплін. 3) Розробити методологію аналізу навчальних програм із застосуванням великих мовних моделей, сформулювати перелік контрольних запитань для тестування. 4) Розробити імплементацію інформаційної технології на базі великих мовних моделей, яка дозволить виконувати аналіз навчальних програм.

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

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ (підпис)

Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Проаналізувати способи видобутку текстової інформації з pdf файлів.*	06.11 – 10.11.2023	
2	*Провести аналіз наявних у відкритому доступі великих мовних моделей, їх API та можливість застосування для задачі видобутку даних із силабусів дисциплін.*	10.11 – 15.11.2023	
3	*Розробити методологію аналізу навчальних програм із застосуванням великих мовних моделей, сформулювати перелік контрольних запитань для тестування.*	15.11 – 25.11.2023	
4	*Розробити імплементацію інформаційної технології на базі великих мовних моделей, яка дозволить виконувати аналіз навчальних програм.*	25.11 – 10.12.2023	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	10.12 – 17.12.2023	

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 55 с., 24 рис., 1 додаток, 36 джерел.

Обґрунтування актуальності теми роботи – Штучний інтелект перебуває на передньому краї розвитку технологій, надаючи значний вплив на різні сфери життя сучасного суспільства, включаючи освіту. Одним із можливих шляхів застосування ШІ є аналіз великих обсягів текстової інформації, виділення фактів та узагальнення. В контексті проблеми, що розглядається, великі мовні моделі, такі як GPT, BART, PEGASUS , T5 мають ряд переваг перед іншими підходами. Вони можуть генерувати більш деталізовану інформацію про зміст навчальних програм, а також можуть використовуватися для виявлення нових тенденцій і закономірностей.

Об’єкт дослідження — застосування великих мовник моделей для аналізу текстів природньою мовою.

Предмет дослідження — Інформаційна технологія порівняння ефективності застосування великих мовних моделей для аналізу текстів природньою мовою на прикладі силабусів дисциплін навчальної програми ОПП «Комп’ютерні науки», освітній ступінь магістр.

Мета роботи — створення інформаційної технології для аналізу навчальних програм дисциплін із застосуванням великих мовних моделей.

Методи дослідження — аналіз моделей та методів аналізу тексту природньою мовою на основі великих мовних моделей.

Результати — Розробити імплементацію інформаційної технології на базі великих мовних моделей, яка дозволить виконувати аналіз навчальних програм, довести її ефективність.

АНАЛІЗ ТЕКСТІВ ПРИРОДНЬОЮ МОВОЮ, ВЕЛИКА МОВНА МОДЕЛЬ, СІЛАБУС, BART ,GPT, PEGASUS, T5

ЗМІСТ

ВСТУП

Актуальність. Нові технології, вимоги ринку праці та інші фактори вимагають від навчальних закладів адаптації своїх навчальних програм задля забезпечення високої якості освіти та конкурентоспроможності випускників. Одним із кроків актуалізації навчальних програм є аналіз змісту дисциплін. Такий аналіз дозволяє отримати інформацію про зміст дисципліни, її цілі та завдання, методи навчання та викладання, а також результати навчання. Ця інформація може бути використана для виявлення можливих проблем у навчальних програмах, а також для розробки рекомендацій щодо їх удосконалення. Аналіз навчальних програм є трудомісткою і часто суб'єктивною процедурою. Велика кількість інформації, що міститься у силабусах та інших документах, робить цей процес важким для виконання вручну.

Об’єкт дослідження. Штучний інтелект перебуває на передньому краї розвитку технологій, надаючи значний вплив на різні сфери життя сучасного суспільства, включаючи освіту. Одним із можливих шляхів застосування ШІ є аналіз великих обсягів текстової інформації, виділення фактів та узагальнення. В контексті проблеми, що розглядається, великі мовні моделі, такі як GPT, BART, PEGASUS , T5 мають ряд переваг перед іншими підходами. Вони можуть генерувати більш деталізовану інформацію про зміст навчальних програм, а також можуть використовуватися для виявлення нових тенденцій і закономірностей. Об’єктом обрано застосування великих мовник моделей для аналізу текстів природньою мовою.

Предмет дослідження. Інформаційна технологія порівняння ефективності застосування великих мовних моделей для аналізу текстів природньою мовою на прикладі силабусів дисциплін навчальної програми ОПП «Комп’ютерні науки», освітній ступінь магістр.

Гіпотеза. Застосування великих мовних моделей дозволить вирішити задачу реферування та узагальнення змісту силабусів, що дозволить

підвищити ефективність оновлення та узгодження цих документів у рамках навчальної програми.

Наукова новизна. Використання великих мовних моделей, таких як GPT, BART, PEGASUS, T5, для аналізу навчальних програм є новаторським підходом. Ці моделі мають потужний аналітичний потенціал, який може бути використаний для отримання детальної інформації з силабусів навчальних дисциплін.

Структура. Кваліфікаційна робота складається з аналітичного огляду способі видобутку текстової інформації з pdf файлів, теоретичних засад побудови великих мовних моделей, аналізу можливостей API відкритих мовних моделей для вирішення задачі видобутку даних із силабусів дисциплін, постановки задачі, розробки методології аналізу ефективності застосування великих мовних моделей для задачі дослідження, імплементації інформаційної технології, висновків, списку використаних жерел, додатку.

1 ОГЛЯД РІШЕНЬ ДЛЯ АНАЛІЗУ ЗМІСТУ ТЕКСТОВОЇ ІНФОРМАЦІЇ ЗА ДОПОМОГОЮ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

1.1 Загальна характеристика трансформерів

Автоматизація обробки текстової інформації входить до галузі обробки природної мови (Natural Language Processing, NLP). NLP визначається як галузь штучного інтелекту, спрямована на розробку методів та моделей для взаємодії комп'ютерів з людською мовою. Ця область має величезний потенціал у багатьох сферах, включаючи автоматизовану обробку текстів, машинний переклад, аналіз настроїв, чат-боти, синтез мовлення та інші завдання, які вимагають розуміння та генерації природної мови. Однією з ключових складових NLP є використання нейронних мереж, які є потужними інструментами для аналізу та обробки текстової інформації. У наш час існує велика кількість різних архітектур та моделей нейронних мереж, призначених для роботи з натуральною мовою, і вони продовжують розвиватися з кожним роком 0.

Трансформер (Transformer) – це революційний підхід до обробки природної мови (Natural Language Processing, NLP). Ця архітектура змінила підхід до завдань обробки тексту і стала ключовим чинником у досягненні вражаючих результатів у багатьох NLP-завданнях, таких як машинний переклад, вибір відповідей, генерація тексту та багато інших. Однією з ключових ідей за архітектурою Transformers є використання механізму уваги. Він дозволяє мережі фокусуватися на різних частинах вхідного тексту під час обробки кожного виходу, не залежно від відстані між словами. Це спрощує моделювання довгострокових залежностей та враховує контекст 0.

Механізм самопідсилювання дозволяє кожному слову в послідовності взаємодіяти з усіма іншими словами в цій послідовності. Це допомагає враховувати важливість кожного слова в контексті, незалежно від його розташування. Transformers можуть використовувати кілька головок

механізму уваги паралельно, що дозволяє їм фокусуватися на різних аспектах контексту. Це підвищує здатність моделі враховувати різні типи залежностей в тексті. Оскільки Transformers не мають вбудованого уявлення про порядок слів у тексті, позиційні ембедінги додають інформацію про позиції слів у послідовності 0.

Transformers можуть використовуватися як енкодери для перетворення вхідного тексту у внутрішнє представлення та як декодери для генерації вихідного тексту, наприклад, у завданнях машинного перекладу. Ця архітектура дозволила покращити результати в багатьох завданнях NLP та вперше надала здатність моделям розуміти багато різних аспектів тексту та його контексту. Transformers стали основою для багатьох сучасних архітектур в галузі NLP і продовжують привертати увагу дослідників і практиків, завдяки їхній ефективності та гнучкості у різних завданнях обробки тексту 0.

Схема архітектури Transformer для обробки природної мови (NLP) (Рисунок 1, Рисунок 1.2): вхідний текст розбивається на токени (слова або підрядки); кожен токен конвертується в векторний формат (наприклад, векторний word embedding) (крок 1); позиційні ембеддинги (Positional Embeddings) додають інформацію про позиції токенів у послідовності; це допомагає моделі розрізняти слова, які входять в різні частини тексту (крок 2); множина енкодерів, кожен з яких використовується для обробки вхідного тексту; кожен енкодер має два основних шари: механізм самопідсилювання (Self-Attention) (дозволяє моделі взаємодіяти між словами в контексті) та Feed-Forward Neural Network (використовується для подальшої обробки інформації) (крок 3); у кожному енкодері механізм самопідсилювання має кілька «голів» (heads), які працюють паралельно; кожна голова навчається визначати різні типи взаємодій між словами (крок 4); результат роботи голів механізму самопідсилювання об'єднується, створюючи заголовок Attention; це внутрішнє представлення тексту з урахуванням контексту (крок 5); додавання з попереднього шару дозволяє моделі легше навчатися глибоким залежностям; нормалізація слідкує за стабільністю навчання (крок 6); енкодери можна

розміщувати в стеку, що дозволяє моделі виявляти більше глибоких залежностей (крок 7); вихід останнього енкодера використовується для подальшого аналізу тексту або для генерації вихідного тексту (наприклад, у завданнях перекладу або генерації тексту) (крок 8); декодери використовуються у завданнях послідовного моделювання (наприклад, машинний переклад); вони містять всі ті ж самі компоненти, що і енкодери, але також мають додатковий механізм завдань і взаємодіють з вихідним текстом (крок 9); вихід останнього декодера може бути використаний для генерації вихідного тексту у відповідь на вхід (крок 10); модель навчається шляхом зворотного поширення помилки (backpropagation) та оновлення ваг моделі під час навчання (крок 11); функція втрат (Loss Function) визначає, наскільки відповідає вихід моделі правильним відповідям; оптимізатор (Optimizer) використовується для оновлення ваг моделі під час навчання (крок 12) 0 – 0.

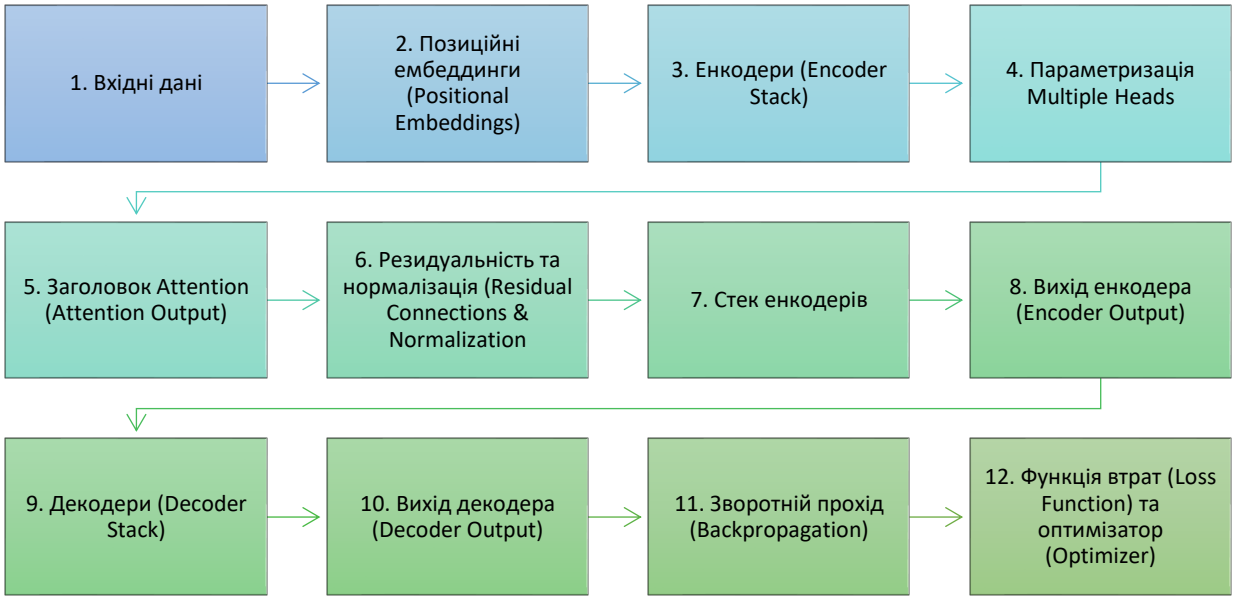

Рисунок 1.1 Алгоритмізована послідовність функціонування Transformer для NLP 0–0

Transformer – це архітектура, яка значно полегшує та покращує обробку природної мови завдяки механізмам самопідсилювання та багатьом іншим інноваційним рішенням.

Трансформер відкрив нові горизонти в глибокому навчанні для обробки природної мови. Однією з його основних переваг є здатність обробки послідовностей будь-якої довжини, що робить його більш гнучким у порівнянні зі звичайними рекурентними нейронними мережами (RNN), де довжина вхідної послідовності може бути обмеженою. Transformer також визначається високою швидкістю навчання, завдяки якій великі мовні моделі можуть бути навчені за прийнятний проміжок часу 00.

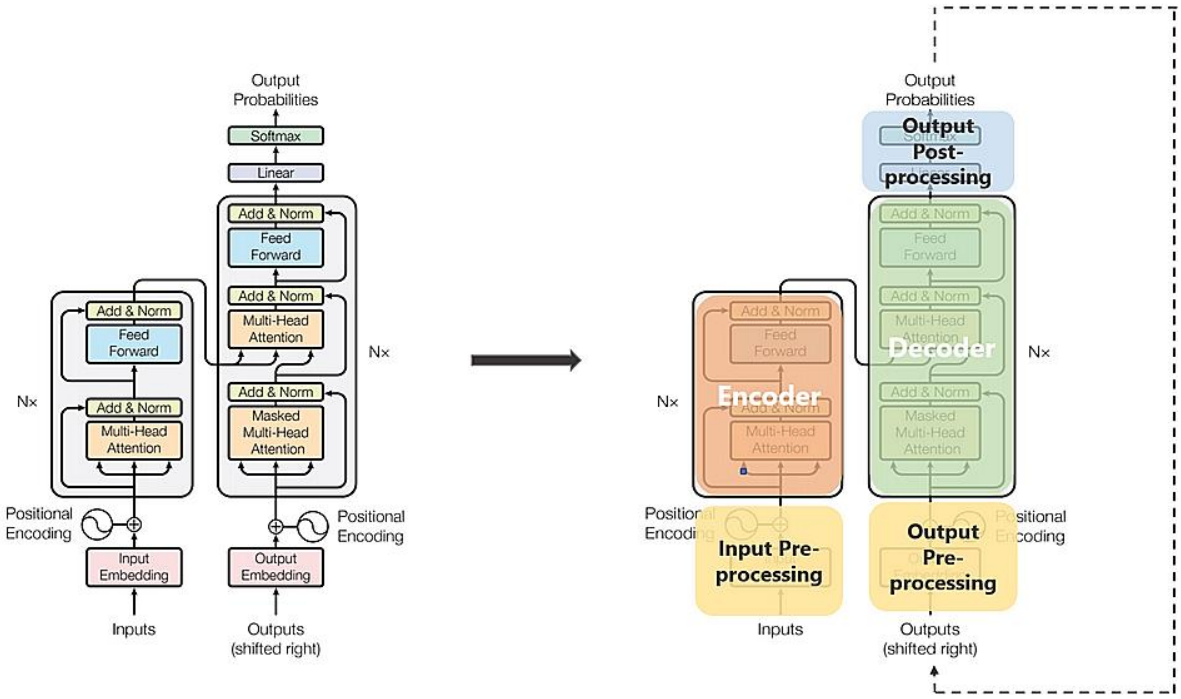

Рисунок 1.2 – Архітектура Transformer для NLP 0 –0

Найбільш відомими представниками нейромереж типу Transformer є GPT та BERT.

Виконаймо порівняння трансформерів з попередніми рішеннями з оброки натуральної мови – Таблиця 1.1.

Таблиця 1.1 – Порівняння характеристики нейромереж при обробці натуральної мови

Характеристика	RNN	CNN	LSTM	Transformer
Обробка послідовностей	Так	Ні	Так	Так
Довжина вхідних послідовностей	Обмежена	Може бути довільною	Обмежена	Може бути довільною
Здатність до урахування контексту	Обмежена	Не враховує контекст	Обмежена	Враховує контекст
Можливість паралельності	Не дуже висока	Висока	Не дуже висока	Висока
Залежність від попередніх кроків	Так	Ні	Так	Ні
Здатність до виявлення довгих залежностей	Обмежена	Обмежена	Добре	Дуже добре
Застосування у завданнях обробки мови	Текстові дані	Текстові та візуальні дані	Текстові дані	Текстові дані, зображення

Аналіз досліджуваних нейромереж та мовних моделей дозволяє визначити переваги та недоліки їх застосування для обробки натуральної мови – Таблиця 1.2.

Таблиця 1.2 – Критичний аналіз нейромереж, що застосовуються для NLP

Нейромережа	Переваги	Недоліки
RNN	Здатність до обробки послідовних даних: RNN придатні для обробки послідовних даних, таких як текст або мовний сигнал. Здатність враховувати контекст: RNN здатні враховувати контекст попередніх кроків в послідовності.	Проблема з довгими залежностями: RNN мають труднощі в виявленні довгих залежностей в даних через проблему з втратою градієнту. Обмежена паралельність: RNN обмежені у паралельній обробці даних, що робить їх менш ефективними для обробки великих обсягів даних.
CNN	Висока паралельність: CNN можуть обробляти багато частин даних паралельно, що робить їх ефективними для обробки зображень та текстів. Здатність до виявлення локальних ознак: CNN здатні виявляти локальні ознаки в даних, що корисно при аналізі текстів.	Неможливість обробки послідовних даних: CNN не завжди підходять для обробки послідовних даних, таких як текст, без використання додаткових методів. Втрата порядку: Вони не здатні враховувати порядок слів у тексті без спеціальних заходів.

Нейромережа	Переваги	Недоліки
LSTM	Здатність до виявлення довгих залежностей: LSTM спроектовані для виявлення та збереження довгих залежностей в даних. Здатність до обробки послідовних даних: Вони підходять для обробки текстових послідовностей та інших послідовних даних.	Вища обчислювальна складність: LSTM вимагають більше обчислювальних ресурсів порівняно з RNN або CNN. Можлива перенавчаність: Вони можуть перенавчатися на довгих послідовностях даних, якщо не враховувати обмеження.
Transformer	Висока паралельність: Transformer є дуже паралельними, що дозволяє їм ефективно обробляти великі обсяги даних. Здатність до виявлення довгих залежностей: Вони добре працюють з довгими послідовностями завдяки механізмам уваги. Здатність до обробки текстових даних: Transformer є потужними при роботі з текстовими даними і завдяки своїм механізмам уваги можуть враховувати порядок слів у тексті.	Вимагає більше даних: Transformer може вимагати більше навчальних даних, щоб працювати ефективно. Вимагає більше обчислювальних ресурсів: Вони вимагають більше обчислювальних ресурсів порівняно з деякими іншими архітектурами.

Загалом, вибір між цими архітектурами залежить від конкретного завдання та обсягу даних. Transformer виявляється дуже потужним у багатьох завданнях обробки мови, але може бути дорожчим за обчислювальними ресурсами. LSTM підходить для завдань з довгими залежностями, RNN може бути використаний для базових завдань, а CNN корисний при обробці тексту та зображень.

1.2 Механізми векторизації та кластеризації текстової інформації

В основі всіх рішень та алгоритмів обробки текстової інформації – механізм кластеризації 0. Наразі існує численна кількість алгоритмів кластеризації, серед яких найбільш вживаним є алгоритм K-means. Однак, наразі, застосовуються більш ефективні сучасні алгоритми кластеризації та векторизації текстової інформації. Розглянемо основні з них.

Word2Vec – це популярний алгоритм у галузі обробки природної мови (NLP), який використовується для векторизації слів у текстах та отримання їхнього семантичного представлення в числовому вигляді. Цей метод дозволяє перетворити слова у вектори у такому просторі, де подібні слова розташовані близько одне до одного, і віддалені слова мають відмінний семантичний зміст. Word2Vec був розроблений компанією Google і став важливим інструментом для роботи з текстовими даними. Word2Vec базується на двох основних архітектурах: Continuous Bag of Words (CBOW) і Skip-gram 0.

Алгоритм Doc2Vec є розширенням алгоритму Word2Vec, спрямованим на векторизацію документів, замість окремих слів чи токенів. Основна ідея Doc2Vec полягає в тому, щоб навчити модель створювати векторні представлення для текстової інформації, які враховують семантику та контекст документа в контексті всього корпусу тексту. Doc2Vec використовує дві архітектурні моделі: Distributed Memory (DM) та Distributed Bag of Words (DBOW) 0.

Top2Vec – це алгоритм групування текстових документів, який використовує методи тематичної моделювання та векторизації документів для автоматичного виявлення семантичних груп або тем в наборі даних. Його основною метою є створення сумісних груп документів, що відображають семантичні концепти або теми 0.

GloVe, або Global Vectors for Word Representation, є алгоритмом для векторизації слів і текстових даних, призначеним для роботи з текстами та розуміння семантики слів у контексті. Він розроблений для створення векторних представлень слів, що можна використовувати в різних задачах обробки природної мови, включаючи групування текстової інформації за змістом 0.

Таким чином, Word2Vec, Doc2Vec, Top2Vec та GloVe – це методи векторизації слів та текстів, які створюють числові представлення для слів чи документів. Вони базуються на статистичних методах та матричних розкладах, а Word2Vec і Doc2Vec також використовують нейромережі.

Трансформери ж, такі як BERT та GPT, використовуються для роботи з текстом та обробки природної мови на більш високому рівні абстракції порівняно з Word2Vec, Doc2Vec, Top2Vec та GloVe. Основна відмінність полягає в тому, що трансформери розуміють семантику тексту, враховуючи контекст та взаємозв'язки між словами у більш складний спосіб.

BERT та GPT – це трансформери, які використовують глибокі нейронні мережі з архітектурою трансформера для розуміння тексту на рівні слів, фраз, речень та контексту в цілому. Вони здатні до передбачення наступних слів у тексті, класифікації текстів за темами, генерації текстів та багатьох інших завдань обробки природної мови 0,0 – Рисунок 1.3.

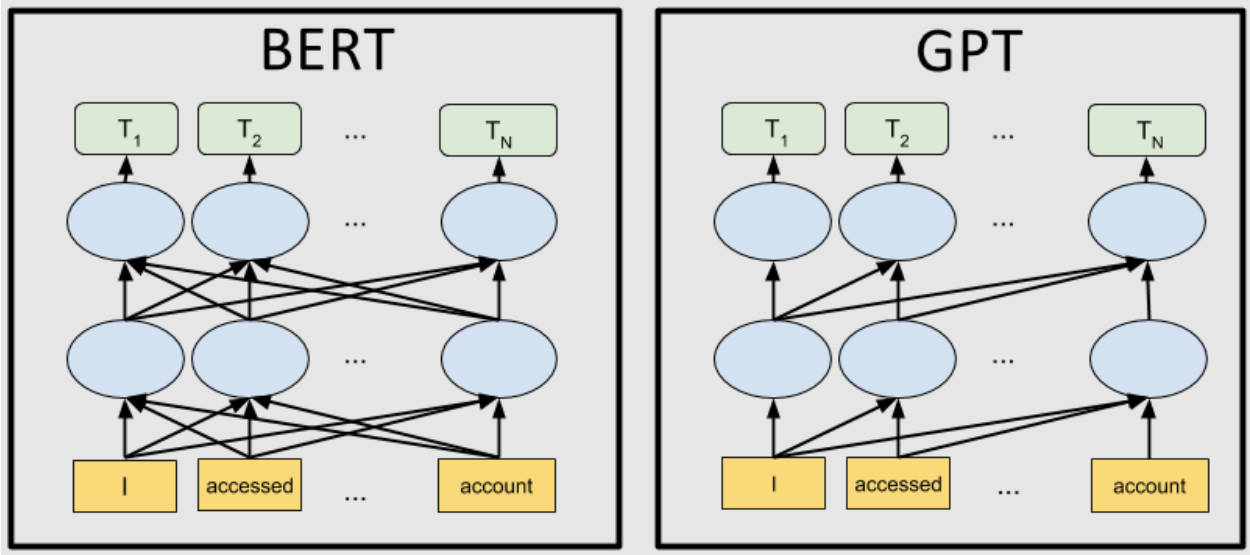

Рисунок 1.3 – Принципові схеми функціонування архітектур трансформер-алгоритмів BERT та GPT 0,0

У реальних застосуваннях може виникати ситуація, коли BERT або GPT використовуються для доопрацювання представлень, створених за допомогою Word2Vec, Doc2Vec, Top2Vec або GloVe. Наприклад, вектори, створені Word2Vec, можуть бути використані як вхідні дані для BERT для вирішення конкретно завдання, такої як класифікація тексту. Такі поєднання дозволяють

використовувати переваги обох підходів: статистичного та контекстно-семантичного моделювання 0,0 – Таблиця 1.3.

Таблиця 1.3 – Компаративний аналіз алгоритмів NLP 0, 0

Характеристика	Word2Vec, Doc2Vec, Top2Vec та GloVe	BERT (BertModel) та GPT (GPT2Model)
Архітектура	Word embeddings	Transformer-based models
Досяжність	Легко доступні і використовуються на практиці	Зазвичай вимагають великої обчислювальної потужності та ресурсів. Можуть бути доступні у вигляді попередньо навчених моделей.
Завдання	Зазвичай використовуються для створення векторних представлень слів та документів, а також для знаходження схожих слів чи документів	Зазвичай використовуються для багатьох завдань обробки природної мови, таких як класифікація тексту, переклад, генерація тексту, розпізнавання іменованих сутностей тощо.
Контекст	Не здатні враховувати довжину або складність текстового контексту	Здатні моделювати багато різних відносин у тексті завдяки трансформер-архітектурі та self-attention механізму.
Навчання	Навчаються на великих текстових корпусах методами, заснованими на статистиці, які не враховують семантику слова	Навчаються на великих текстових корпусах, враховуючи семантику та контекст слів, завдяки чому здатні до кращого розуміння тексту.
Застосування	Зазвичай використовуються для векторизації слів та документів, а також для простого групування текстових даних	Застосовуються в багатьох областях обробки природної мови, зокрема в аналізі сентименту, машинному перекладі, автоматичній підтримці клієнтів тощо.
Структура моделі	Не мають внутрішньої структури, що розуміє контекст або граматику	Мають глибоку ієрархічну структуру з численними шарами, які розуміють семантику та граматику тексту.
Здатність до самонавчання	Не здатні до навчання під час використання	Можуть піддаватися до навчання під час використання на нових даних.
Швидкість навчання	Відносно швидке	Вимагає більше часу для навчання через складність моделі.

Загалом, Word2Vec, Doc2Vec, Top2Vec, GloVe – це методи для створення векторних представлень слів та документів та їх використання для

деяких задач обробки природної мови. У той час як BERT та GPT – це більш потужні та глибокі моделі, які здатні розуміти семантику та контекст тексту та вирішувати більш складні завдання обробки природної мови. Вибір методу залежить від конкретної задачі та доступних ресурсів.

1.3 Визначення оптимальних трансформерів для вирішення завдання з аналізу текстової інформації

Внаслідок аналізу існуючих методів та моделей у галузі обробки природної мови на сучасному етапі можна зробити висновок, що трансформери є найкращими рішеннями в даній області. Трансформери – це архітектурні парадигми та моделі глибокого навчання, які значно покращили результати багатьох завдань NLP, зокрема машинного перекладу, сентимент-аналізу, кластеризації тексту, та інших 0.

Головні переваги трансформерів включають в себе вміння моделі ефективно моделювати довгі залежності між словами в тексті завдяки механізму уваги, який дозволяє враховувати контекст та зв'язки між словами. Вони є більш ефективними, ніж рекурентні нейронні мережі (RNN), для яких вони були спочатку розроблені. Трансформи також є більш гнучкими, ніж згорткові нейронні мережі (CNN), і можуть бути використані для вирішення широкого спектру завдань NLP. Крім того, трансформери легко паралелізуються, що робить їх ефективними на сучасних обчислювальних пристроях, включаючи графічні процесори (GPU) та тензорні процесори (TPU) 0.

Важливо зазначити, що трансформери також стали основою для багатьох важливих моделей в NLP, таких як BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer) та інші 0. Загалом наразі налічується близько 238 трансформерів від різних розробників 0 – Таблиця 1.4.

Таблиця 1.4 – Аналіз існуючих трансформерів для NLP 0

Трансформери			
ALBERT	DPR	MaskFormer	SegFormer
XLNet	YOSO	CPM	DPR
BigBird-RoBERTa	ByT5	DePlot	FLAN-T5
BERTweet	Jukebox	GPT-3.5	GPT-4
XLM-V	BARThez	BARTpho	PhoBERT
XLS-R	LLaMA2	NLLB	Segment Anything
ALIGN	DPT	MaskFormerSwin	SEW
AltCLIP	EfficientFormer	mBART	SEW-D
Audio Spectrogram Transformer	EfficientNet	MEGA	Speech Encoder decoder
Autoformer	ELECTRA	Megatron-BERT	Speech2Text
Bark	EnCodec	MGP-STR	Speech2Text2
BART	Encoder decoder	MobileBERT	SpeechT5
BEiT	ERNIE	MobileNetV1	Splinter
BERT	ErnieM	MobileNetV2	SqueezeBERT
Bert Generation	ESM	MobileViT	SwiftFormer
BigBird	FairSeq Machine-Translation	MobileViTV2	Swin Transformer
BigBird-Pegasus	Falcon	MPNet	Swin Transformer V2
BioGpt	FlauBERT	MPT	Swin2SR
BiT	FLAVA	MRA	SwitchTransformers
Blenderbot	FNet	MT5	T5
BlenderbotSmall	FocalNet	MusicGen	Table Transformer
BLIP	Funnel Transformer	MVP	TAPAS
DialoGPT	DiT	LayoutXLM	TAPEX
BLIP-2	GIT	NAT	Time Series Transformer
BLOOM	GLPN	Nezha	TimeSformer
BridgeTower	GPT Neo	NLLB-MOE	TimmBackbone
CamemBERT	GPT NeoX	Nyströmformer	Trajectory Transformer
CANINE	GPT NeoX Japanese	OneFormer	Transformer-XL
Chinese-CLIP	GPT-J	OpenAI GPT	TrOCR
CLAP	GPT-Sw3	OpenAI GPT-2	TVLT
FLAN-UL2	MatCha	T5v1.1	UL2
CLIP	GPTBigCode	OpenLlama	UMT5
CLIPSeg	GPTSAN-japanese	OPT	UniSpeech
CodeGen	Graphormer	OWL-ViT	UniSpeechSat
CodeLlama	GroupViT	Pegasus	UPerNet
Conditional DETR	Hubert	PEGASUS-X	VAN
ConvBERT	I-BERT	Perceiver	VideoMAE
ConvNeXT	IDEFICS	Pix2Struct	ViLT

ConvNeXTV2	ImageGPT	PLBart	Vision Encoder decoder
CPM-Ant	Informer	PoolFormer	VisionTextDualEncoder
CTRL	InstructBLIP	Pop2Piano	VisualBERT
CvT	Jukebox	ProphetNet	ViT
Data2VecAudio	LayoutLM	PVT	ViT Hybrid
Data2VecText	LayoutLMv2	QDQBert	VitDet
Data2VecVision	LayoutLMv3	RAG	ViTMAE
DeBERTa	LED	REALM	ViTMSN
DeBERTa-v2	LeViT	Reformer	VITS
Decision Transformer	LiLT	RegNet	ViViT
Deformable DETR	LLaMA	RemBERT	Wav2Vec2
DeiT	Longformer	ResNet	Wav2Vec2-Conformer
DETA	LongT5	RetriBERT	WavLM
DETR	LUKE	RoBERTa	Whisper
DiNAT	LXMERT	RoBERTa-PreLayerNorm	X-CLIP
DistilBERT	M2M100	RoFormer	XGLM
DonutSwin	Marian	RWKV	XLM
XLM-RoBERTa	MarkupLM	SAM	XLM-ProphetNet
mBART-50	MMS	Wav2Vec2Phoneme	XLM-RoBERTa-XL
DINOv2	M-CTC-T	RoCBert	X-MOD
XLM-RoBERTa-XL	Mask2Former	XLNet	YOLOS
Whisper	Megatron-GPT2		
Ключ			
	Google		Facebook/Meta
	Tsinghua University		VinAI
	Microsoft		Salesforce
	OpenAI		SHI Labs
	EleutherAI		NVIDIA
	Berkeley		HuggingFace

Відповідно до результатів аналізу номенклатури існуючих трансформерів (Таблиця 1.4) визначаємо, що провідними розробниками технології NLP є Google, Facebook/Meta, OpenAI, Microsoft та NVIDIA.

Разом з тим наведений датасет вказує на можливість визначення головних гілок еволюційного розвитку трасформерів 0 – Таблиця 1.5.

Таблиця 1.5 – Аналіз походження технології трансформерів 0

Трансформери			
ALBERT	DPR	MaskFormer	SegFormer
ALIGN	DPT	MaskFormerSwin	SEW
AltCLIP	EfficientFormer	mBART	SEW-D
Audio Spectrogram Transformer	EfficientNet	MEGA	Speech Encoder decoder
Autoformer	ELECTRA	Megatron-BERT	Speech2Text
Bark	EnCodec	MGP-STR	Speech2Text2
BART	Encoder decoder	MobileBERT	SpeechT5
BEiT	ERNIE	MobileNetV1	Splinter
BERT	ErnieM	MobileNetV2	SqueezeBERT
Bert Generation	ESM	MobileViT	SwiftFormer
BERTweet	Jukebox	GPT-3.5	GPT-4
BigBird	FairSeq Machine-Translation	MobileViTV2	Swin Transformer
BigBird-Pegasus	Falcon	MPNet	Swin Transformer V2
BigBird-RoBERTa	ByT5	DePlot	FLAN-T5
BioGpt	FlauBERT	MPT	Swin2SR
BiT	FLAVA	MRA	SwitchTransformers
Blenderbot	FNet	MT5	T5
BlenderbotSmall	FocalNet	MusicGen	Table Transformer
BLIP	Funnel Transformer	MVP	TAPAS
BLIP-2	GIT	NAT	Time Series Transformer
BLOOM	GLPN	Nezha	TimeSformer
BridgeTower	GPT Neo	NLLB-MOE	TimmBackbone
CamemBERT	GPT NeoX	Nyströmformer	Trajectory Transformer
CANINE	GPT NeoX Japanese	OneFormer	Transformer-XL
Chinese-CLIP	GPT-J	OpenAI GPT	TrOCR
CLAP	GPT-Sw3	OpenAI GPT-2	TVLT
CLIP	GPTBigCode	OpenLlama	UMT5
CLIPSeg	GPTSAN-japanese	OPT	UniSpeech
CodeGen	Graphormer	OWL-ViT	UniSpeechSat
CodeLlama	GroupViT	Pegasus	UPerNet
Conditional DETR	Hubert	PEGASUS-X	VAN
ConvBERT	I-BERT	Perceiver	VideoMAE
ConvNeXT	IDEFICS	Pix2Struct	ViLT
ConvNeXTV2	ImageGPT	PLBart	Vision Encoder decoder
CPM-Ant	Informer	PoolFormer	VisionTextDualEncoder
CTRL	InstructBLIP	Pop2Piano	VisualBERT
CvT	Jukebox	ProphetNet	ViT

Data2VecAudio	LayoutLM	PVT	ViT Hybrid
Data2VecText	LayoutLMv2	QDQBert	VitDet
Data2VecVision	LayoutLMv3	RAG	ViTMAE
DeBERTa	LED	REALM	ViTMSN
DeBERTa-v2	LeViT	Reformer	VITS
Decision Transformer	LiLT	RegNet	ViViT
Deformable DETR	LLaMA	RemBERT	Wav2Vec2
DeiT	Longformer	ResNet	Wav2Vec2-Conformer
DETA	LongT5	RetriBERT	WavLM
DETR	LUKE	RoBERTa	Whisper
DialoGPT	DiT	LayoutXLM	TAPEX
DiNAT	LXMERT	RoBERTa-PreLayerNorm	X-CLIP
DINOv2	M-CTC-T	RoCBert	X-MOD
DistilBERT	M2M100	RoFormer	XGLM
DonutSwin	Marian	RWKV	XLM
FLAN-UL2	MatCha	T5v1.1	UL2
mBART-50	MMS	Wav2Vec2Phoneme	XLM-RoBERTa-XL
Whisper	Megatron-GPT2		
XLM-RoBERTa	MarkupLM	SAM	XLM-ProphetNet
XLM-RoBERTa-XL	Mask2Former	XLNet	YOLOS
XLM-V	BARThez	BARTpho	PhoBERT
XLNet	YOSO	CPM	DPR
XLS-R	LLaMA2	NLLB	Segment Anything
Ключ			
BERT	BART	GPT	Transformer

Відповідно до аналізу походження технології трансформерів (Таблиця 1.5), спостерігаємо, що базовими NLP-моделями є BERT, BART, GPT, Transformer, що узгоджується з загальним уявленням про еволюцію даного виду ШІ 0 – Рисунок 1.4.

Разом з тим спостерігаємо, що NLP-трансформери постійно удосконалюються, збільшуючи при цьому у розмірах 0 – Рисунок 1.5.

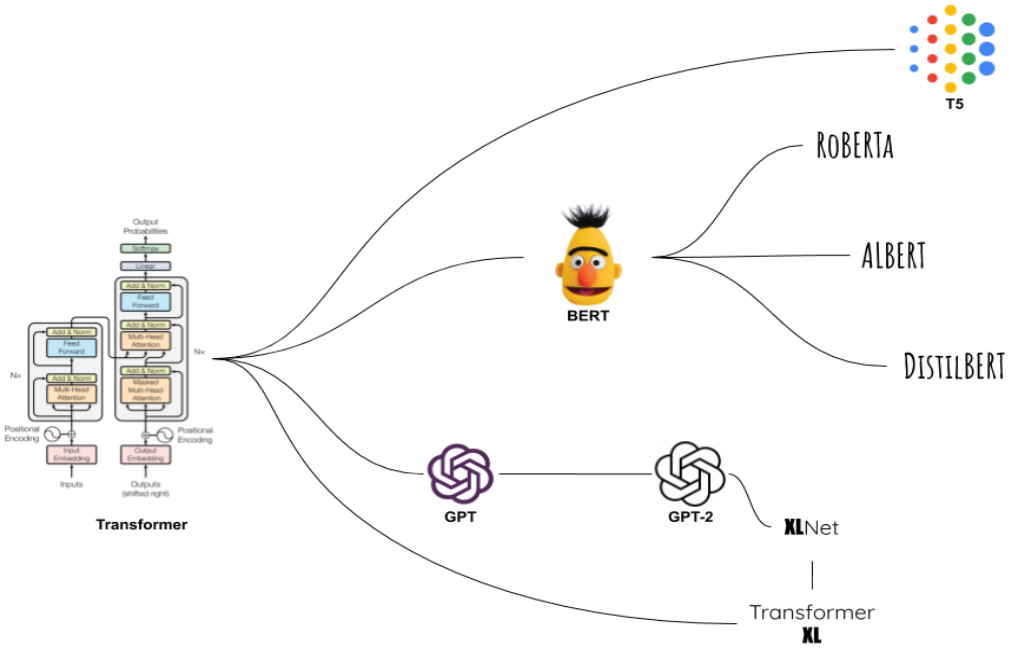

Рисунок 1.4 – Головні гілки еволюційного розвитку NLP-трансформерів 0

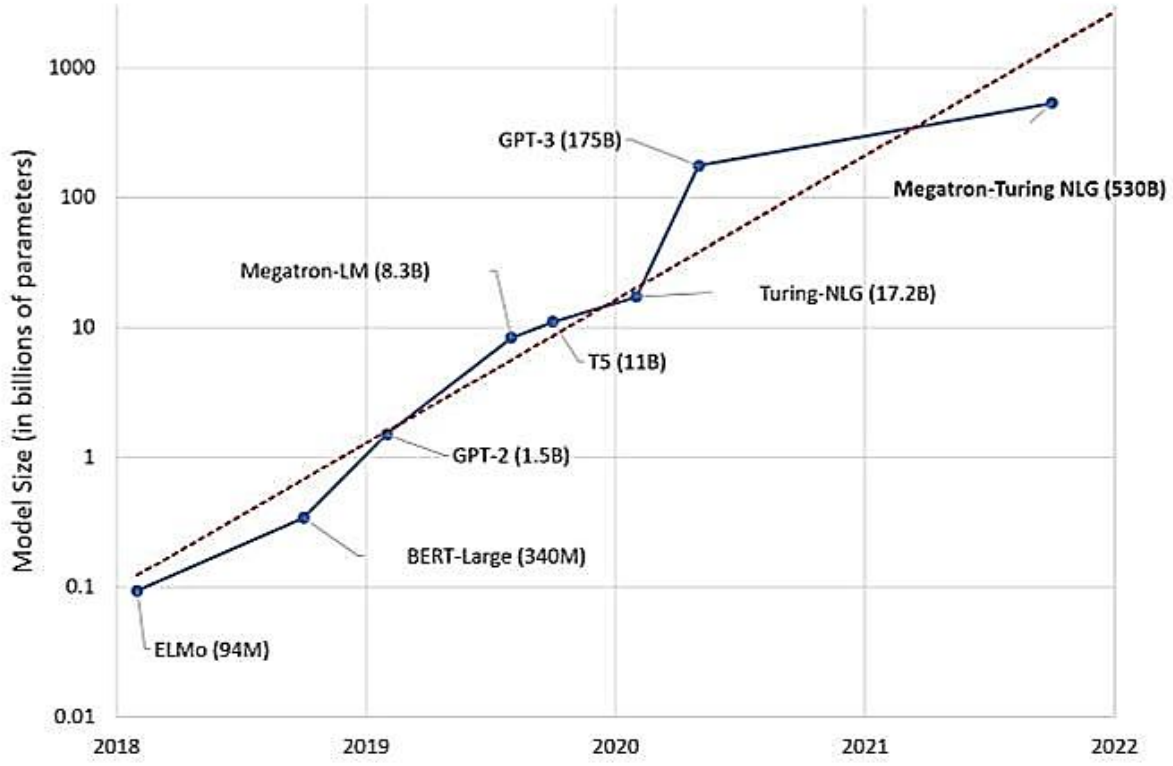

Рисунок 1.5 – Хронометрична динаміка розміру трансформерів для NLP 0

Більші мовні моделі є більш точними, однак зі збільшення параметризації збільшується і вимогливість до ресурсної та обчислювальної бази. Відповідно необхідно балансувати між точністю, ефективністю та

витратністю. Саме оптимальний баланс функціонування трансформеру визначає його застосовуваність 0 – Рисунок 1.6.

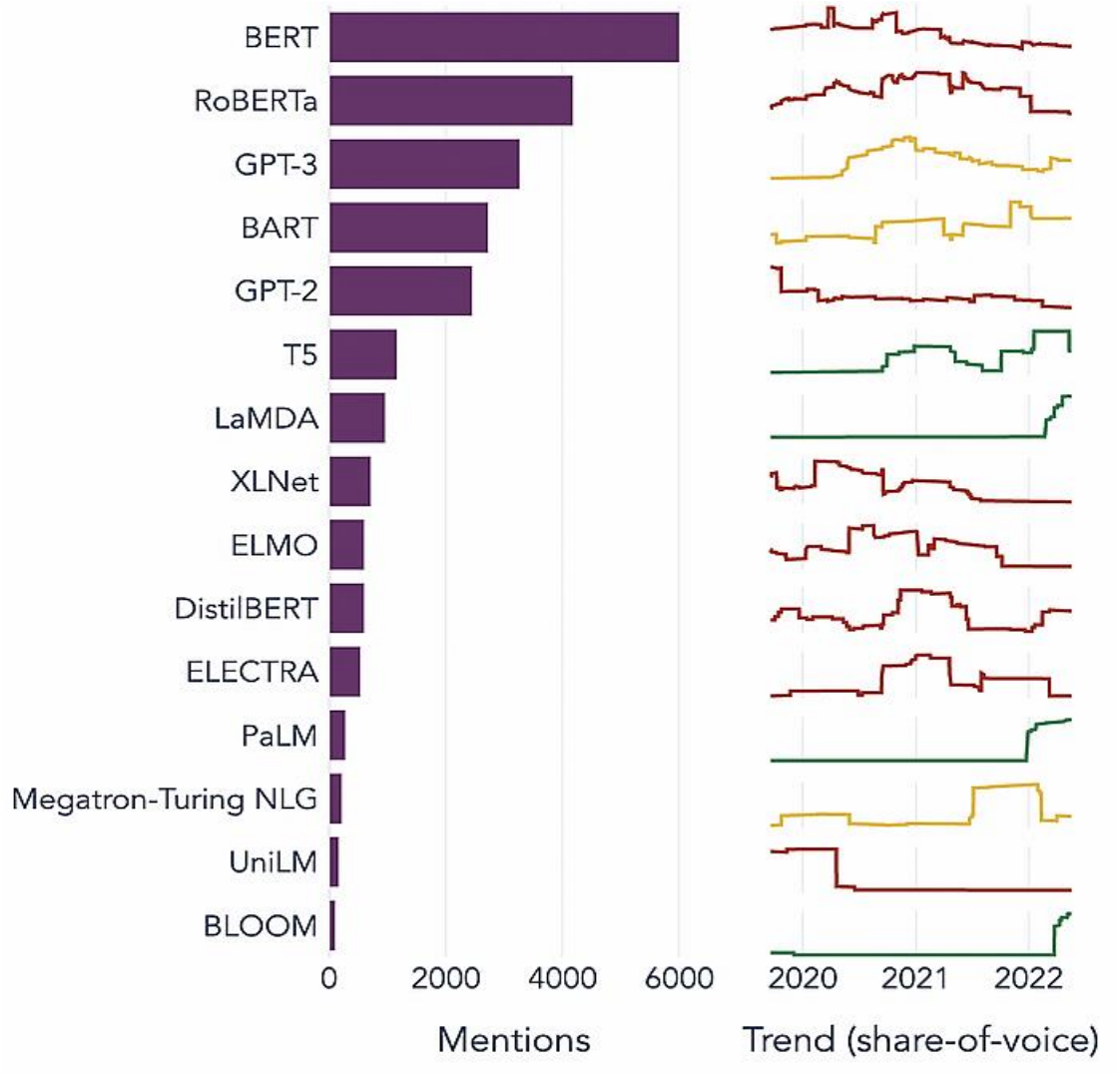

Рисунок 1.6 – Популярність трансформерів у експертному середовищі 0

Оскільки досліджувані трансформери мовні моделі, то до них застосовується відповідна система параметричних критеріїв, що дозволяє ранжувати дані технології за загальною ефективністю 0 – Рисунок 1.7.

Model	Core differentiator	Pre-training objective	Para-meters	Access	Information Extraction	Text Classification	Conversa-tional AI	Summari-zation	Machine Translation	Content generation
BERT	First transformer-based LLM	AE	370M	Source code						
RoBERTa	More robust training procedure	AE	354M	Source code						
GPT-3	Parameter size	AR	175B	API						
BART	Novel combination of pre-training objectives	AR and AE	147M	Source code						
GPT-2	Parameter size	AR	1.5B	Source code						
T5	Multi-task transfer learning	AR	11B	Source code						
LaMDA	Dialogue; safety and factual grounding	AR	137B	No access						
XLNet	Joint AE and AR	AE and AR	110M	Source code						
DistilBERT	Reduced model size via knowledge distillation	AE	82M	Source code						
ELECTRA	Computational efficiency	AE	335M	Source code						
PaLM	Training infrastructure	AR	540B	No access						
MT-NLG	Training infrastructure	AR and AE	530B	API						
UniLM	Optimised both for NLU and NLG	Seq2seq, AE and AR	340M	Source code						
BLOOM	Multilingual (46 languages)	AR	176B	Source code						

AR = Autoregression
AE = Autoencoding
Seq2seq = Sequence-to-sequence

Highly appropriate
Appropriate
Somewhat appropriate

Рисунок 1.7 – Параметричний аналіз трансформерів як моделей обробки натуральної мови 0

Відповідно до результатів аналізу досліджуваних трансформерів за системою параметричних критеріїв (Рисунок 1.7) найефективнішими є мовні моделі на основі технології BERT та GPT, що відповідає сучасним тенденціям, однак більшість експертів вважають, що розробка Google більш придатна для класифікації та обробки тексту, а розробка OpenAI – для генерації та контекстуальної семантики 0– 0.

1.4 Постановка задачі дослідження

Сучасний вищий освітній процес вимагає високоточних інструментів для аналізу та вдосконалення освітніх програм. У цьому контексті велике значення має аналіз освітнього змісту силабусів, які є ключовими документами, що визначають структуру та зміст навчальних курсів. З метою забезпечення ефективного та систематичного аналізу силабусів, виникає потреба в розробці програмного рішення, що здатне автоматично проводити цей процес.

Метою цього проєкту є створення програмного рішення, яке надасть можливість автоматичного аналізу освітнього змісту силабусів з використанням передових технологій обробки природної мови. Програма буде призначена для ефективного виділення ключових елементів, визначення структури та семантики тексту, а також для забезпечення корисної інформації для подальших аналітичних завдань.

Задачі:

1. Розробити алгоритм автоматичного аналізу силабусів, зокрема виділення основних розділів, ключових термінів та інших важливих компонентів.
2. Інтегрувати трансформерні моделі для роботи з текстовим матеріалом та отримання більш глибокого розуміння семантики силабусу.
3. Реалізувати можливість створення автоматичних звітів або підсумкових документів на основі аналізу, що полегшить розуміння змісту курсу.

Визначення вимог до програмного рішення є ключовим етапом у процесі розробки, оскільки воно визначає функціональні та нефункціональні характеристики, які вирішують проблеми користувачів та гарантують успішність проєкту.

Вимоги до програмного рішення з автоматичного аналізу освітнього змісту силабусів з використанням трансформерів 00:

1. Загальні вимоги:

1.1. Програмне рішення призначене для автоматизованого аналізу освітнього змісту силабусів.

1.2. Має забезпечувати ефективне виділення ключових елементів та семантики тексту силабусу.

2. Функціональні вимоги:

2.1. Аналіз текстового матеріалу:

2.1.1. Виділення основних розділів, тем та підрозділів у силабусі.

2.1.2. Розпізнавання ключових термінів та фраз, характерних для кожного розділу.

2.1.3. Оцінка обсягу та важливості кожного розділу в контексті силабусу.

2.2. Використання трансформерів:

2.2.1. Інтеграція трансформерних моделей для аналізу семантики тексту.

2.2.2. Визначення структури та взаємозв'язків між ключовими елементами.

2.3. Створення звітів та підсумкових документів:

2.3.1. Генерація автоматичних звітів на основі аналізу силабусу.

2.3.2. Можливість експорту звітів у різноманітні формати (текст, PDF, HTML тощо).

3. Інтерфейс:

3.1. Користувацький інтерфейс:

3.1.1. Інтуїтивно зрозумілий та простий для використання інтерфейс.

3.1.2. Можливість завантаження силабусу у різних форматах (PDF, DOCX, TXT).

4. Надійність та продуктивність:

4.1. Програма повинна працювати стабільно при обробці силабусів різного обсягу та складності.

4.2. Забезпечити швидку обробку текстового матеріалу та аналіз за розумний час.

5. Забезпечення безпеки:

5.1. Забезпечення конфіденційності та безпеки завантажених силабусів.

5.2. Захист від можливих атак, включаючи віруси та шкідливі програми.

6. Міжплатформенність та масштабованість:

6.1. Робота програми на різних операційних системах (Windows, macOS, Linux).

6.2. Здатність програми працювати з великою кількістю силабусів одночасно.

7. Технічна підтримка:

7.1. Наявність системи підтримки користувачів.

7.2. Регулярні оновлення програмного забезпечення для виправлення помилок та вдосконалення функціоналу.

Ці вимоги визначають фундаментальні критерії для успішної реалізації програмного рішення з аналізу освітнього змісту силабусів.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО РІШЕННЯ ДЛЯ АНАЛІЗУ ЗМІСТУ ТЕКСТОВОЇ ІНФОРМАЦІЇ ЗА ДОПОМОГОЮ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

2.1 Аналіз та вибір способів навчання моделі

BERT (Bidirectional Encoder Representations from Transformers) – це попередньо навчена модель глибокого нейронного мережевого архітектурного типу, яка використовує трансформери для завдань обробки природної мови (NLP). Модель BERT була навчена на великому корпусі тексту, щоб здобути глибокі та універсальні репрезентації слів та фраз. Процес pre-training включає в себе навчання моделі на великому обсязі тексту без конкретних завдань аналізу або класифікації 0.

Використання попередньо підготовленої моделі BERT у додатках NLP охоплює структурований процес, що включає кілька ключових кроків 0 (Рисунок):

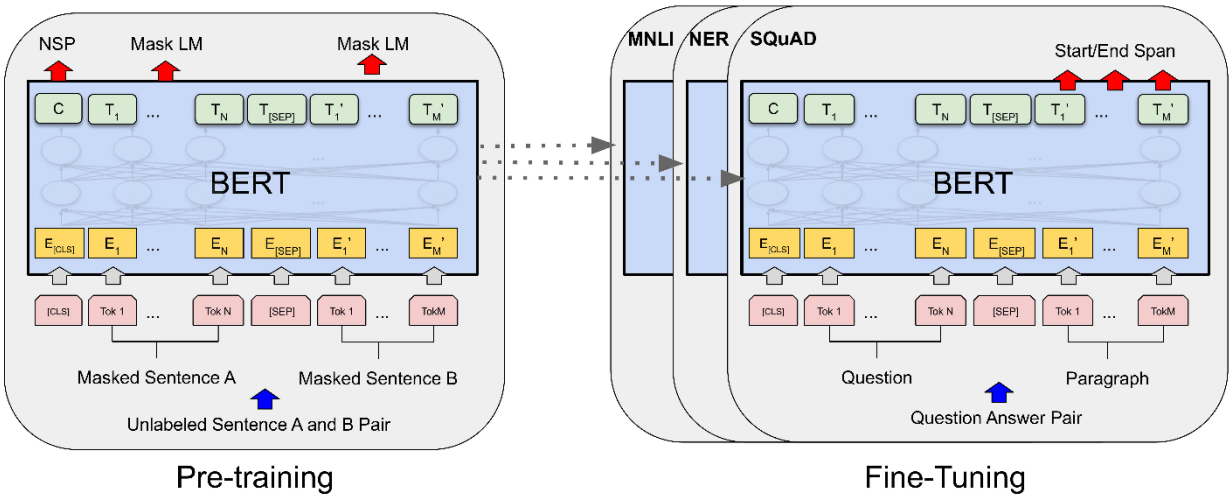

Рисунок 2.1 – Загальний процес використання pre-training BERT 0

1. Початковий крок передбачає вибір відповідної попередньо підготовленої моделі BERT на основі мовних вимог і характеру

поставленого завдання. Доступні моделі BERT, попередньо навчені різноманітним текстовим корпусам, мовам і доменам, що забезпечує гнучкість у виборі моделі.

2. Після вибору моделі обрана попередньо навчена модель BERT піддається архітектурним модифікаціям, щоб адаптувати її до конкретного завдання. Ця адаптація часто тягне за собою додавання шарів або заголовків, специфічних для завдань, до базової архітектури BERT. Ці додаткові компоненти служать інтерфейсами для обробки унікальних характеристик завдання.
3. Підготовка навчальних даних є критичною фазою процесу. Дані мають бути відформатовані та структуровані відповідно до вхідних вимог модифікованої моделі BERT. Це включає токенізацію, доповнення та кодування текстових даних, що забезпечує їх ефективну обробку моделлю.
4. Коли модель налаштована та дані підготовлені, починається етап тонкої настройки (Fine-Tuning). Під час тонкого налаштування адаптована модель BERT додатково навчається на наборі даних для конкретного завдання. Цей набір даних, як правило, анотований мітками основної істини або цілями, що мають відношення до конкретного завдання NLP. Тонка настройка дозволяє моделі вивчати шаблони, нюанси та особливості, що стосуються конкретного завдання.
5. Ітерації тонкого налаштування включають кілька епох навчання. Модель ітеративно уточнює свої параметри шляхом мінімізації визначеної функції втрат, яка кількісно визначає розбіжність між прогнозованими результатами та цільовими показниками наземної істини. Навчання триває до конвергенції або до тих пір, поки конкретні показники ефективності не вкажуть на задовільну продуктивність.

6. Протягом усього процесу тонкого налаштування важливо відстежувати продуктивність моделі на окремому наборі даних перевірки. Показники оцінки, специфічні для завдання, такі як accuracy, F1-score, або perplexity, використовуються для вимірювання ефективності моделі. Це гарантує, що модель добре узагальнюється та уникає переобладнання.
7. Гіперпараметри (Hyperparameter), включно зі швидкістю навчання, розмірами пакетів і показниками вилучення, оптимізовані для підвищення продуктивності моделі. Це може включати проведення пошуку гіперпараметрів для визначення найбільш ефективних конфігурацій.
8. Після завершення тонкого налаштування адаптована модель BERT розгортається для висновку щодо нових, невідомих даних. Модель приймає вхідний текст і створює прогнози або виходи, адаптовані до конкретного завдання, наприклад класифікація тексту, розпізнавання іменованих об’єктів або машинний переклад.
9. Процес часто включає ітераційні цикли уточнення, коли вносяться коригування в архітектуру моделі, попередню обробку даних або стратегію тонкого налаштування на основі інформації, отриманої в результаті оцінки моделі та реальної продуктивності.

Таким чином, продемонстровано загальний алгоритм використання попередньо навченого трансформера BERT, попередній навчальний досвід якого можливо адаптувати під конкретизовані завдання з класифікації тексту.

2.2 Use-case діаграми програмного рішення

Розробка Use-Case діаграми для програмного рішення з автоматичного аналізу освітнього змісту силабусів з використанням трансформерів є

ключовим етапом в процесі проектування. Ця діаграма визначає функціональність системи та взаємодію між різними акторами та випадками використання 0.

Ця діаграма допомагає визначити основні функції, які система має виконувати, і визначити, як різні користувачі (актори) будуть взаємодіяти з програмним рішенням. Вона висвітлює сценарії використання, які служитимуть основою для подальшого розроблення та тестування системи.

Актори системи (Рисунок 2.2):

1. Користувач:
   - Основний актор, що взаємодіє з програмою.
   - Має можливість завантажувати силабуси для аналізу та перегляду результатів.
2. Система:
   - Використовує трансформери для проведення аналізу текстового матеріалу силабусів.
   - Забезпечує вивід результатів та статистики користувачеві.

Випадки використання:

1. Завантаження силабусу:
   - Користувач завантажує силабус у програму для подальшого аналізу.
2. Аналіз силабусу:
   - Система використовує трансформери для аналізу текстового матеріалу силабусу.
   - Визначення ключових термінів, тем, та їх взаємозв'язків.
3. Вивід результатів:
   - Система генерує звіт та виводить результати аналізу користувачеві.

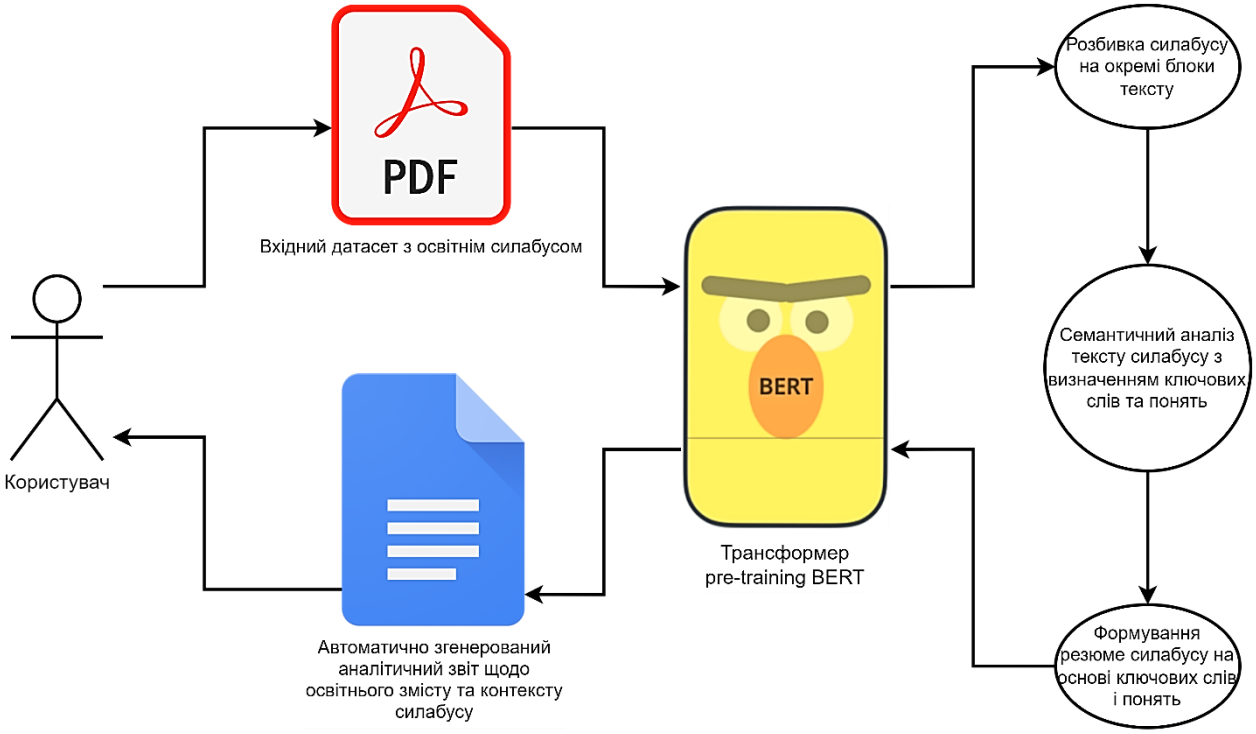

Рисунок 2.2 – Use-Case діаграма програмного рішення з автоматичного аналізу освітнього змісту силабусів з використанням трансформерів

Створення цієї діаграми допомагає команді розробників та зацікавленим сторонам легше розуміти, як програмне рішення буде використовуватися в реальному середовищі. Це важливий інструмент для спілкування із замовниками, командою розробників та іншими учасниками проекту, щоб забезпечити високий рівень зрозуміння функціоналу системи та очікуваних результатів.

2.3 Вибір програмних засобів та інструментів для розробки

У якості програмного середовища для реалізації проєктного програмного рішення обираємо мову Python 0.

Використання мови програмування Python для розробки програмного рішення з аналізу освітнього змісту силабусів за допомогою трансформерів обумовлене кількома об'єктивними факторами. По-перше, Python

визначається простотою та легкістю читання коду, що полегшує процес розробки та обслуговування програми. По-друге, наявність різноманітних бібліотек для обробки тексту, зокрема TensorFlow, PyTorch і Transformers, спрощує імплементацію та використання трансформерів для аналізу текстової інформації. Крім того, активна спільнота Python забезпечує доступність документації та високий рівень підтримки, що сприяє зручній розробці та уникненню потенційних труднощів.

Серед фрейморків, що можна обрати для розробки проєктного програмного продукту – PyTorch, TensorFlow та Flax/JAX. PyTorch, TensorFlow та Flax/JAX – це фреймворки для машинного навчання та глибокого навчання, які надають інструменти для створення, навчання і розгортання нейронних мереж та моделей глибокого навчання.

У якості фреймворку для розробки проєктного програмного продукту обираємо PyTorch 0. Вибір фреймворку PyTorch для реалізації програмного рішення з автоматичного аналізу освітнього змісту силабусів з використанням трансформерів ґрунтується кількома обґрунтованими перевагами. По-перше, PyTorch є потужним та гнучким інструментом для роботи з нейромережами, забезпечуючи простоту розробки та ефективність використання різноманітних архітектур, включаючи трансформери. По-друге, активна спільнота користувачів PyTorch та наявність широкого спектру ресурсів для навчання та підтримки дозволяють розробникам швидко освоювати нові можливості та експериментувати з архітектурними рішеннями. Крім того, PyTorch надає зручний інтерфейс для роботи з передньо-навченими моделями, такими як трансформери, спрощуючи процес адаптації цих моделей до конкретних завдань аналізу освітніх програм. Такий вибір фреймворку сприяє високій ефективності та легкості розробки програмного рішення для обробки та аналізу текстової інформації в силабусах.

Головним інструментом у розробці проєктного програмного рішення є бібліотека transformers 4.33.3 від Hugging Face 0.

Бібліотека Transformers (раніше відома як pytorch-transformers і pytorch-pretrained-bert) надає сучасні загальнопризначені архітектури (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, CTRL...) для розуміння природної мови (Natural Language Understanding, NLU) та генерації природної мови (Natural Language Generation, NLG). У бібліотеці наявно понад 32 передньо-навчені моделі в більш ніж 100 мовах і глибока взаємодія між TensorFlow 2.0 та PyTorch.

Основні можливості та характеристики бібліотеки transformers 4.33.3 0–0:

1. Бібліотека надає легкий доступ до широкого спектру передньо-навчених моделей, таких як BERT, GPT-2, RoBERTa, T5, та багатьох інших. Вона дозволяє завантажувати ці моделі та використовувати їх для завдань обробки тексту.
2. transformers 4.33.3допомагає токенізувати текст, підготувати вхідні дані для моделей та обробляти результати.
3. Бібліотека дозволяє використовувати передньо-навчені моделі для фінетюнінгу на конкретних завданнях, таких як класифікація тексту, генерація тексту, витягнення інформації тощо.
4. transformers 4.33.3 надає засоби для генерації тексту за допомогою мовних моделей. Вона дозволяє створювати текст, перекладати мови, генерувати діалоги та інше.
5. transformers 4.33.3 підтримує багато мов, що дозволяє використовувати моделі для різноманітних мовних завдань.
6. Бібліотека має активну спільноту користувачів і розробників, що сприяє постійному розвитку та оновленню бібліотеки.

Загалом, бібліотека transformers 4.33.3 є потужним інструментом для розробки та використання моделей глибокого навчання в області NLP і знаходить застосування в багатьох областях, включаючи обробку тексту, машинний переклад, автоматичну генерацію тексту, аналіз настроїв та багато інших завдань.

При застосуванні попередньо навчених мовних моделей відсутня необхідність у розробці спеціальної бази. Код завантажує попередньо надану модель BERT для обробки тексту та використовує її для автоматичного аналізу освітнього змісту силабусів та збереження відповідного резюме результатів у файл. Всі операції виконуються в пам'яті та не вимагають спеціальної бази даних.

3 ТЕСТУВАННЯ ПРОГРАМНОГО РІШЕННЯ

3.1 Аналіз функціонування проєктних програмних рішень

Виконаємо аналіз функціонування проєктного програмного застосунку для автоматичного аналізу освітнього змісту силабусів з використанням трансформерів.

Імпорт бібліотек (Рисунок 3.1). Імпортуємо необхідні бібліотеки, включаючи pdfplumber для вилучення тексту з PDF і BertModel та BertTokenizer з бібліотеки transformers для аналізу BERT. Імпортуємо бібліотеку torch для операцій з тензорами.

```
import pdfplumber
from transformers import BertModel, BertTokenizer
import torch
```

Рисунок 3.1 – Імпорт необхідних бібліотек

Визначення функції для вилучення тексту з PDF (Рисунок). Створюємо функцію extract_text_from_pdf, яка приймає шлях до файлу як вхідний параметр, відкриває PDF за допомогою pdfplumber і вилучає текст з кожної сторінки.

```
def extract_text_from_pdf(file_path):
    with pdfplumber.open(file_path) as pdf:
        text = ""
        for page in pdf.pages:
            text += page.extract_text()
    return text
```

Рисунок 3.2 – Визначення функції для вилучення тексту з PDF

Визначення функції для аналізу блоку силабусу з використанням BERT (Рисунок). Створюємо функцію analyze_syllabus_block, яка приймає текстовий блок, модель BERT та токенізатор BERT в якості вхідних параметрів. Токенізуємо блок, подаємо його через модель BERT, визначаємо топ-5 ключових слів на основі індексів токенів. Розраховуємо середнє значення векторів токенів для представлення понять. Повертаємо словник, який містить ключові слова, поняття та зведену інформацію.

```
def analyze_syllabus_block(block, bert_model, bert_tokenizer):
    tokenized_text = bert_tokenizer(block, return_tensors="pt", truncation=True, max_length=512)
    output = bert_model(**tokenized_text)[0]
    top_k_values, top_k_indices = torch.topk(output, k=5, dim=-1)
    keywords = [bert_tokenizer.decode(token.item()) for token in top_k_indices.flatten()]
    concepts = output.mean(dim=0).tolist()
    summary = {
        "keywords": keywords,
        "concepts": concepts,
        "summary": " ".join(keywords),
    }
    return summary
```

Рисунок 3.3 – Визначення функції для аналізу блоку силабусу з використанням BERT

Визначення функції для аналізу всього силабусу (Рисунок 3.4). Створюємо функцію analyze_syllabus, яка приймає весь текст силабусу, модель BERT та токенізатор BERT в якості вхідних параметрів. Розбиваємо текст на блоки. Застосовуємо функцію analyze_syllabus_block до кожного блоку та зберігаємо результат у списку. Повертаємо список зведених інформацій по блоках.

Визначення функції для виведення зведення (Рисунок). Створюємо функцію print_summary, яка приймає результат аналізу (список зведених інформацій по блоках) та виводить інформацію для кожного блоку. Виводимо ключові слова, поняття та роздільник між блоками.

```
def analyze_syllabus(text, bert_model, bert_tokenizer):
    blocks = text.split("\n\n")
    summary = [analyze_syllabus_block(block, bert_model, bert_tokenizer) for block in blocks]
    return summary
```

Рисунок 3.4 – Визначення функції для аналізу всього силабусу

```
def print_summary(summary):
    for item in summary:
        print("**Keywords:**", ", ".join(item["keywords"]))
        print("**Concepts (mean of token embeddings):**", item["concepts"])
        print("**Summary:**", item["summary"])
        print("\n" + "="*40 + "\n")
```

Рисунок 3.5 – Визначення функції для виведення зведення

Визначення функції для завантаження моделі BERT та токенізатора (Рисунок 8). Створюємо функцію load_bert_model_and_tokenizer, яка завантажує модель BERT та токенізатор з переднавченої версії "bert-base-uncased". Повертає завантажену модель та токенізатор.

```
def load_bert_model_and_tokenizer():
    bert_model = BertModel.from_pretrained("bert-base-uncased")
    bert_tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
    return bert_model, bert_tokenizer
```

Рисунок 8 – Визначення функції для завантаження моделі BERT та токенізатора

Виконання основної програми (Рисунок). Перевіряємо, чи сценарій виконується безпосередньо (if __name__ == "__main__":). Завантажуємо модель BERT та токенізатор. Намагаємося вилучити текст з вказаного файлу PDF, проаналізувати силабус та вивести зведену інформацію. Обробляємо винятки та виводимо повідомлення про помилку у разі їх виникнення.

```
if __name__ == "__main__":
    pdf_file_path = "syllabus.pdf"

    try:
        bert_model, bert_tokenizer = load_bert_model_and_tokenizer()

        syllabus_text = extract_text_from_pdf(pdf_file_path)
        analysis_result = analyze_syllabus(syllabus_text, bert_model, bert_tokenizer)
        print_summary(analysis_result)

    except Exception as e:
        print(f"Error: {e}")
```

Рисунок 3.7 – Виконання основної програми

Цей код виконує екстракцію тексту з PDF, токенізацію за допомогою BERT та створення зведення для кожного блоку силабусу на основі аналізу BERT. Створене зведення містить ключові слова, поняття (представлені середнім значенням векторів токенів), і сформатоване зведення. Код розроблено для обробки помилок та виведення інформативних повідомлень у випадку виникнення помилок.

3.2 Тестування проєктних програмних рішень

У якості вхідного датасету використовуємо «Data Science Fundamentals with Python and Excel» з освітнього ресурсу СумДУ 0 – Рисунок 3.8, Рисунок .

Рисунок 3.8 – Тестувальний вхідний датасет 0

SYLLABUS

1. General information on the course

Full course name	Data Science Fundamentals with Python and Excel
Full official name of a higher education institution	Sumy State University
Full name of a structural unit	Faculty of Electronics and Informational Technologies. Кафедра комп'ютерних наук
Author(s)	Tyrkusova Nadiia Volodymyrivna
Cycle/higher education level	The First Level Of Higher Education, National Qualifications Framework Of Ukraine – The 6th Level, QF-LLL – The 6th Level, FQ-EHEA – The First Cycle
Semester	16 weeks across 4 semester
Workload	5 ECTS, 150 hours. For full-time course 48 hours are working hours with the lecturer (16 hours of lectures, 32 hours of laboratory work), 102 hours of the individual study.
Language(s)	English

2. Place in the study programme

Relation to curriculum	Elective course available for the students of the specialty 122 "Computer Sciences"
Prerequisites	Necessary knowledge of general disciplines: "Higher Mathematics"; and professional training "Programming", "Organization and processing of electronic information"
Additional requirements	There are no specific requirements
Restrictions	There are no specific restrictions

Рисунок 3.9 – Короткий зміст тестувального силабусу 0

Відповідно до логічного алгоритму проєктного програмного застосунку з автоматичного аналізу освітнього змісту силабусів з застосуванням трансформерів отримали відповідне резюме – Рисунок 3.10, Рисунок 3.11.

Короткий резюме силабусу "Data Science Fundamentals with Python and Excel"

Загальна інформація:

- **Повна назва:** "Основи науки про дані з Python та Excel"
- **Установа:** Сумський державний університет
- **Факультет:** Електроніки та інформаційних технологій
- **Кафедра:** Комп'ютерних наук
- **Автор(ка):** Тиркусова Надія Володимирівна
- **Цикл/рівень вищої освіти:** Перший цикл вищої освіти, Національна рамка кваліфікацій України – 6-й рівень, QF-LLL – 6-й рівень, FQ-EHEA – Перший цикл
- **Семестр:** 16 тижнів, 4 семестрових тижні
- **Обсяг навантаження:** 5 кредитів ECTS, 150 годин. Для очної форми навчання 48 годин – аудиторна робота з викладачем (16 годин лекцій, 32 години лабораторних робіт), 102 години – самостійна робота
- **Мова навчання:** Англійська

Мета курсу:

- Досягнення студентами сучасного конструктивного, фундаментального рівня мислення, а також володіння комплексом методів аналізу даних для задач зв'язку, діагностики та управління процесами та системами різного призначення за допомогою можливостей Excel та Python.

Зміст:

- **Вступ.** Статистика, наука про дані, математичні моделі.
- **Детальна статистика.** Загальна сукупність, вибірка та вимоги до неї. Емпіричний закон розподілу. Асиметрія, куртоз. Статистична оцінка параметрів вибірки.
- **Статистичне перевіряння гіпотез.** Основні поняття та визначення статистичної теорії прийняття рішень. Перевірка гіпотез стосовно закону розподілу.
- **Аналіз зв'язку між змінними.** Кореляційний аналіз. Типи зв'язків між випадковими величинами. Кореляція.
- **Регресійний аналіз.** Парний регресійний аналіз.

Очікувані результати навчання:

- Розраховувати та аналізувати основні числові характеристики вибірки. На основі аналізу параметрів вибірки висувати та перевіряти гіпотезу закону розподілу генеральної сукупності.
- Використовувати статистичне перевіряння гіпотез для вирішення практичних задач.
- Перевіряти зв'язок між факторами за допомогою кореляційного аналізу.
- Будувати стохастичні моделі процесів та явищ, оцінювати параметри моделей та перевіряти їхню значимість. Оцінювати якість та адекватність отриманих моделей.
- Використовувати можливості Python та Excel для аналізу статистичних даних.

Навчально-методичні заходи:

- Лекції, лабораторні роботи, самостійна робота, консультації.

Рисунок 3.10 – Резюме тестувального силабусу

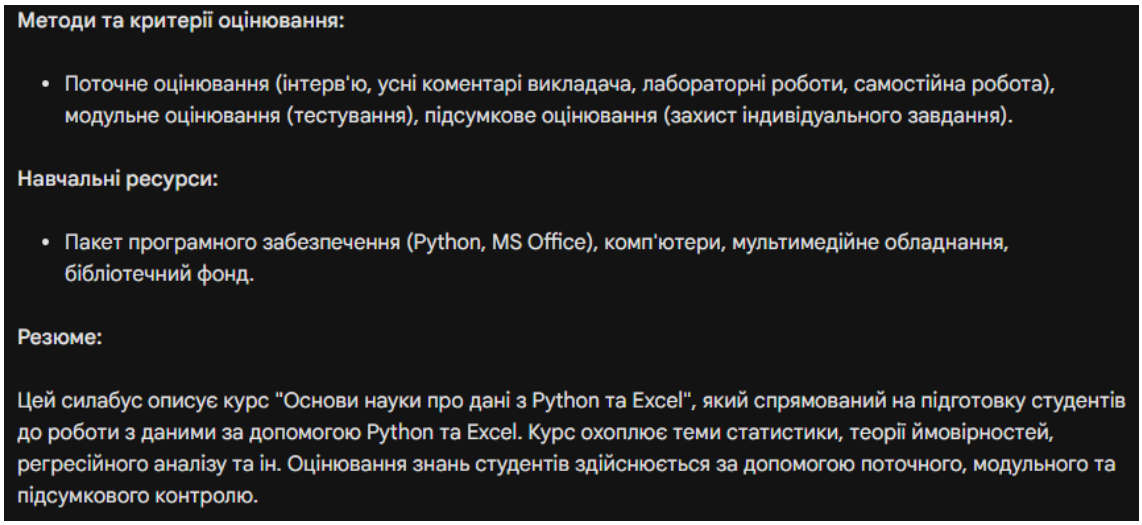

Рисунок 3.11 – Резюме тестувального силабусу (продовження)

Таким чином, програмний застосунок на основі мовної моделі pre-training BERT дозволив виконати автоматичну обробку тексту тестового силабусу СумДУ та сформувати відповідне інформативне резюме, яке є структурованим і дозволяє отримати змістовне уявлення про освітній контекст, що закладається у даному навчальному регламенті. Відповідно підтверджена функціональність та працездатність проєктного програмного рішення.

3.3 Компаративний аналіз ефективності вирішення завдання з аналізу змісту текстової інформації навчальних дисциплін з використанням оптимальних трансформерів

Бібліотека transformers 4.33.3 від Hugging Face 0 дозволяє використовувати не лише pre-training BERT. Як зазначено в описі (п. 2.3 поточної роботи), у бібліотеці наявно понад 32 передньо-навчені моделі, серед яких також є GPT-2, RoBERTa, XLM, DistilBert, XLNet, CTRL та інші.

Відповідно до висновків, щодо визначення оптимальних трансформерів для вирішення завдання з аналізу текстової інформації (п. 1.3 поточної роботи), встановлено, що такими є BERT та GPT.

Зважаючи на виявлені обставини, доцільно для визначених великих мовних моделей виконати компаративний аналіз ефективності вирішення завдання з аналізу змісту текстової інформації навчальних дисциплін.

Інтегруємо відповідний трансформер у проєктний код – Рисунок .

```
import pdfplumber
from transformers import GPTModel, GPTTokenizer
import torch
```

Рисунок 3.12 – Заміна мовної моделі

Виконуємо аналіз того самого датасету «Data Science Fundamentals with Python and Excel» з освітнього ресурсу СумДУ 0 – Рисунок 3.8, Рисунок . Результати виконання робочого циклу проєктного застосунку інварійованого трансформером GPT наводяться нижче – Рисунок 3.13.

Резюме силабусу "Основи науки про дані з Python та Excel"

1. **Загальна інформація про курс:**
   - Назва курсу: Основи науки про дані з Python та Excel.
   - Університет: Сумський державний університет, Факультет електроніки та інформаційних технологій, Кафедра комп'ютерних наук.
   - Автор: Тиркусова Надія Володимирівна.
   - Рівень вищої освіти: Перший рівень вищої освіти, 6-й рівень кваліфікаційної рамки України.
   - Тривалість: 16 тижнів протягом 4 семестрів.
   - Обсяг навантаження: 5 ECTS, 150 годин, зокрема 48 годин лекцій та 102 години індивідуальної роботи.
2. **Місце в навчальній програмі:**
   - Відношення до навчальної програми: Вибірковий курс для студентів спеціальності 122 "Комп'ютерні науки".

Рисунок 3.13 – Резюме тестувального силабусу за допомогою GPT

- Вимоги до попередніх знань: Висока математика, Програмування, Організація та обробка електронної інформації.

3. **Мета курсу:**
   - Розвинення конструктивного, фундаментального рівня мислення у студентів.
   - Оволодіння комплексом методів аналізу даних для завдань управління, діагностики та керування процесами і системами.
4. **Зміст:**
   - Вступ. Статистика, наука про дані, математичні моделі.
   - Описова статистика.
   - Статистичне тестування гипотез.
   - Перевірка взаємозв'язку між змінними. Кореляційний аналіз.
   - Регресійний аналіз.
5. **Очікувані результати навчання:**
   - Розраховувати та аналізувати основні числові характеристики вибірки.
   - Використовувати статистичне тестування гіпотез для розв'язання практичних завдань.
   - Перевіряти взаємозв'язок між факторами за допомогою кореляційного аналізу.
   - Будувати стохастичні моделі процесів та явищ, оцінювати параметри моделей та перевіряти їхню значущість.
   - Використовувати засоби Python та Excel для аналізу статистичних даних.
6. **М'які навички:**
   - Вміння використовувати інформаційно-комунікаційні технології.
   - Абстрактне мислення, аналіз та синтез.
   - Здатність пошуку, обробки та аналізу інформації з різних джерел.
   - Здатність застосовувати знання в практичних ситуаціях.
   - Здатність приймати обґрунтовані рішення.
   - Здатність комунікувати у державній мові, як усно, так і письмово.
7. **Методи навчання:**
   - Лекції, орієнтовані на практику та самостійну роботу.
   - Інтерактивні лекції.
   - Виконання лабораторних завдань.
   - Підготовка лабораторних звітів та їх захист.
   - Виконання і захист індивідуальних завдань.

Рисунок 9.14 – Резюме тестувального силабусу за допомогою GPT (продовження)

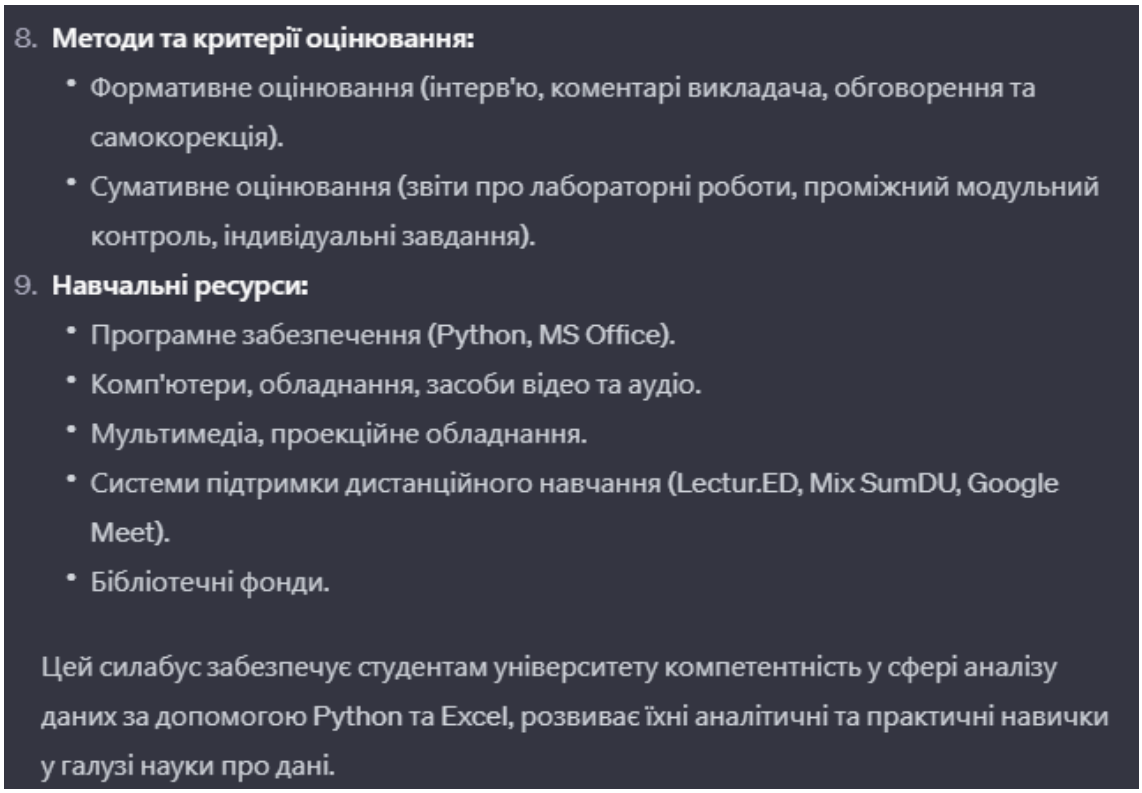

Рисунок 3.15 – Резюме тестувального силабусу за допомогою GPT (продовження)

Відповідно, спостерігаємо що трансформери та GPT успішно виконали відповідний семантичний аналіз наданої текстової інформації.

При цьому отримали певні відмінності:

- Резюме згенероване програмним застосунком на основі BERT надає докладнішу інформацію про установу, факультет, кафедру, автора, цикл вищої освіти, тривалість семестру, обсяг навантаження, мову навчання та мету курсу. Зміст курсу подано у вигляді тем і підтем, а також вказано очікувані результати навчання.
- Резюме згенероване програмним застосунком на основі GPT, хоча містить загальну інформацію, є менш деталізованим у висвітленні деяких аспектів, таких як тривалість семестру та обсяг навантаження. Однак воно вкладає акцент на рівні вищої освіти, вимогах до попередніх знань, методах навчання та оцінюванні, включаючи м'які навички, які студенти розвиватимуть.

Обидва резюме містять загальну мету курсу, зміст та очікувані результати навчання. Резюме BERT може вважатися більш структурованим та деталізованим, в той час як резюме GPT забезпечує більше акценту на загальних аспектах курсу та розвитку м'яких навичок.

Компаративний аналіз ефективності вирішення завдання з аналізу змісту текстової інформації навчальних дисциплін за використання оптимальних трансформерів, таких як BERT і GPT, вказує на їхні відмінності в підходах та можливостях.

BERT, використаний для генерації резюме 1, відзначається високою точністю у вирішенні завдань, пов'язаних із структурованою інформацією, такою як назва курсу та обсяг навантаження. Його сильна сторона полягає в можливості надавати конкретну та деталізовану інформацію.

Натомість GPT, що використовується для генерації резюме 2, відрізняється у вмінні створювати більш загальний текст та фокусуватися на ключових аспектах, таких як мета курсу та очікувані результати. Він проявляє себе як інструмент для створення загальних та концептуальних описів.

Отже, вибір між BERT і GPT залежить від конкретних вимог завдання: BERT надає точну інформацію, тоді як GPT витягує ключові аспекти та створює загальний опис.

ВИСНОВОК

У відповідності до поставленої мети та завдань в даному досліджені отримані наступні рішення:

- встановлено, що наразі найбільш доцільним інструментом для автоматичної обробки і аналізу текстової інформації в рамках технології NLP є великі мовні моделі – трансформери;
- серед провідних трасформерів у якості основного обрано мовну модель BERT;
- програмний застосунок для автоматичного аналізу освітнього змісту в силабусах виконаний на мові програмування Python у фреймворці PyTorch з застосуванням бібліотеки transformers 4.33.3 від Hugging Face;
- тестування на даних силабусу СумДУ довело функціональність та працездатність проєктного програмного застосунку;
- для аналізу можливих варіацій застосування великих мовних моделей, виконана варіація програмного рішення з застосуванням конкуруючого трансформеру GPT. Компаративний аналіз ефективності використання трансформерів BERT і GPT для аналізу навчальних дисциплін підкреслює їхні особливості. BERT, застосований для генерації варіації резюме силабусу 1, відзначається високою точністю в обробці структурованої інформації та здатністю надавати конкретні деталі. З іншого боку, GPT, використаний для варіації резюме силабуса, проявляється як більш загальний, акцентуючи увагу на ключових аспектах та концептуальних описах. Вибір між ними буде залежати від конкретних вимог завдання: BERT найкраще підходить для точного аналізу даних, тоді як GPT відмінно впорається зі створенням загальних та концептуальних описів навчальних дисциплін.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Praful Bharadiya J. A comprehensive survey of deep learning techniques natural language processing. European journal of technology. 2023. Vol. 7, no. 1. P. 58–66. URL: https://doi.org/10.47672/ejt.1473 (date of access: 20.09.2023).

2. Transformers: state-of-the-art natural language processing / T. Wolf et al. Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations, Online. Stroudsburg, PA, USA, 2020. URL: https://doi.org/10.18653/v1/2020.emnlp-demos.6 (date of access: 21.09.2023).

3. Von der Mosel J., Trautsch A., Herbold S. On the validity of pre-trained transformers for natural language processing in the software engineering domain. IEEE transactions on software engineering. 2022. P. 1. URL: https://doi.org/10.1109/tse.2022.3178469 (date of access: 21.09.2023).

4. Wolf T., Tunstall L., Werra L. V. Natural language processing with transformers, revised edition. O'Reilly Media, Incorporated, 2022. URL: https://www.oreilly.com/library/view/natural-language-processing/9781098136789/ (date of access: 21.09.2023).

5. Kjell O., Giorgi S., Schwartz H. A. The text-package: an r-package for analyzing and visualizing human language using natural language processing and transformers. Psychological methods. 2023. URL: https://doi.org/10.1037/met0000542 (date of access: 21.09.2023).

6. Patwardhan N., Marrone S., Sansone C. Transformers in the real world: A survey on NLP applications. Information. 2023. Vol. 14, no. 4. P. 242. URL: https://doi.org/10.3390/info14040242 (date of access: 21.09.2023).

7. Rahali A., Akhloufi M. A. End-to-End Transformer-Based Models in Textual-Based NLP. Ai. 2023. Vol. 4, no. 1. P. 54–110. URL: https://doi.org/10.3390/ai4010004 (date of access: 21.09.2023).

8. Pinheiro W., Fernandes R., Souza L. Thematic grouping for messages in major events. International journal of information systems and project management.

2022. Vol. 4, no. 4. P. 51–65. URL: https://doi.org/10.12821/ijispm040403 (date of access: 21.09.2023).

9. Cooperative perception: mitigating messages content duplication / B. S. Chawky et al. 2022 5th international conference on communications, signal processing, and their applications (ICCSPA), Cairo, Egypt, 27–29 December 2022. 2022. URL: https://doi.org/10.1109/iccspa55860.2022.10019115 (date of access: 21.09.2023).

10. Detection of temporal shifts in semantics using local graph clustering / N. Hwang et al. Machine learning and knowledge extraction. 2023. Vol. 5, no. 1. P. 128–143. URL: https://doi.org/10.3390/make5010008 (date of access: 21.09.2023).

11. Zhu J.-J., Ren Z. J. The evolution of research in resources, conservation & recycling revealed by Word2vec-enhanced data mining. Resources, conservation and recycling. 2023. Vol. 190. P. 106876. URL: https://doi.org/10.1016/j.resconrec.2023.106876 (date of access: 21.09.2023).

12. Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec / D. Kim et al. Information sciences. 2019. Vol. 477. P. 15–29. URL: https://doi.org/10.1016/j.ins.2018.10.006 (date of access: 21.09.2023).

13. Egger R., Yu J. A topic modeling comparison between LDA, NMF, top2vec, and bertopic to demystify twitter posts. Frontiers in sociology. 2022. Vol. 7. URL: https://doi.org/10.3389/fsoc.2022.886498 (date of access: 21.09.2023).

14. Defect texts mining of secondary device in smart substation with glove and attention-based bidirectional LSTM / K. Chen et al. Energies. 2020. Vol. 13, no. 17. P. 4522. URL: https://doi.org/10.3390/en13174522 (date of access: 21.09.2023).

15. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt / C. Zhou et al. arXiv preprint arXiv:2302.09419. 2023. URL: https://doi.org/10.48550/arXiv.2302.09419 (date of access: 21.09.2023).

16. Trummer I. From bert to gpt-3 codex: harnessing the potential of very large language models for data management. arXiv preprint arXiv:2306.09339. 2023. URL: https://doi.org/10.14778/3554821.3554896 (date of access: 21.09.2023).

17. Improving text classification with transformer / G. Soyalp et al. 2021 6th international conference on computer science and engineering (UBMK), Ankara, Turkey, 15–17 September 2021. 2021. URL: https://doi.org/10.1109/ubmk52708.2021.9558906 (date of access: 26.09.2023).

18. Abstractive Text Summarization for Resumes With Cutting Edge NLP Transformers and LSTM / Ö. B. Mercan et al. arXiv preprint arXiv:2306.13315. 2023. URL: https://doi.org/10.48550/arXiv.2306.13315 (date of access: 26.09.2023).

19. Rothman D. Transformers for natural language processing: build, train, and fine-tune deep neural network architectures for NLP with python, pytorch, tensorflow, BERT, and GPT-3. Packt Publishing, Limited, 2022. URL: https://www.packtpub.com/product/transformers-for-natural-language-processing/9781800565791 (date of access: 26.09.2023).

20.Transformers. PyPI.URL: https://pypi.org/project/transformers/#description (date of access: 26.09.2023).

21. An intuitive explanation of transformer-based models. Factored | Machine Learning, Data Engineering and Data Analytics Company. URL: https://factored.ai/transformer-based-language-models/ (date of access: 26.09.2023).

22. Choosing the right language model for your NLP use case. Medium. URL: https://towardsdatascience.com/choosing-the-right-language-model-for-your-nlp-use-case-1288ef3c4929 (date of access: 26.09.2023).

23. Singh J. Natural Language Processing in the Real World: Text Processing, Analytics, and Classification. CRC Press. 2023. URL: https://doi.org/10.1201/9781003264774 (date of access: 26.09.2023).

24. Recent progress on text summarisation based on BERT and GPT / B. Yang et al. Knowledge science, engineering and management. Cham, 2023. P. 225–241. URL: https://doi.org/10.1007/978-3-031-40292-0_19 (date of access: 26.09.2023).

25. Text summarization for big data analytics: A comprehensive review of GPT 2 and BERT approaches / G. Bharathi Mohan et al. Internet of things. Cham, 2023. P. 247–264. URL: https://doi.org/10.1007/978-3-031-33808-3_14 (date of access: 26.09.2023).

26. Silva Barbon R., Akabane A. T. Towards transfer learning techniques–bert, distilbert, bertimbau, and distilbertimbau for automatic text classification from different languages: a case study. Sensors. 2022. Vol. 22, no. 21. P. 8184. URL: https://doi.org/10.3390/s22218184 (date of access: 27.09.2023).

27. Onan A. Hierarchical graph-based text classification framework with contextual node embedding and bert-based dynamic fusion. Journal of king saud university - computer and information sciences. 2023. P. 101610. URL: https://doi.org/10.1016/j.jksuci.2023.101610 (date of access: 27.09.2023).

28. Kassab M. H., Laplante P. A. Requirements engineering for software and systems. Auerbach Publishers, Incorporated, 2022. URL: https://doi.org/10.1201/9781003129509 (date of access: 28.09.2023).

29. Sherif E., Helmy W., Galal-Edeen G. H. Proposed framework to manage non-functional requirements in agile. IEEE access. 2023. P. 1. URL: https://doi.org/10.1109/access.2023.3281195 (date of access: 28.09.2023).

30. A different approach on automated use case diagram semantic assessment / R. Fauzan et al. International journal of intelligent engineering and systems. 2021. Vol. 14, no. 1. P. 496–505. URL: https://doi.org/10.22266/ijies2021.0228.46 (date of access: 28.09.2023).

31. Welcome to Python.org. Python.org. URL: https://www.python.org/ (date of access: 28.09.2023).

32. PyTorch. PyTorch. URL: https://pytorch.org/ (date of access: 28.09.2023).

33. transformers 4.33.3. State-of-the-art Machine Learning for JAX, PyTorch and TensorFlow by Hugging Face. PyPI. URL: https://pypi.org/project/transformers/#description (date of access: 28.09.2023).

34. Jain S. M. Hugging face. Introduction to transformers for NLP. Berkeley, CA, 2022. P. 51–67. URL: https://doi.org/10.1007/978-1-4842-8844-3_4 (date of access: 28.09.2023).

35. Pourkeyvan A., Safa R. Sorourkhah, A.Harnessing the Power of Hugging Face Transformers for Predicting Mental Health Disorders in Social Networks. arXiv preprint arXiv:2306.16891. 2023. URL: https://doi.org/10.48550/arXiv.2306.16891 (date of access: 28.09.2023).

36. Каталог курсів. Особистий кабінет СумДУ. URL: https://pg.cabinet.sumdu.edu.ua/catalog (дата звернення: 12.12.2023).

ДОДАТОК А. Лістинг програмного коду

```
import pdfplumber
from transformers import BertModel, BertTokenizer
import torch

def extract_text_from_pdf(file_path):
    with pdfplumber.open(file_path) as pdf:
        text = ""
        for page in pdf.pages:
            text += page.extract_text()
    return text

def analyze_syllabus_block(block, bert_model, bert_tokenizer):
    tokenized_text = bert_tokenizer(block, return_tensors="pt", truncation=True, max_length=512)
    output = bert_model(**tokenized_text)[0]
    top_k_values, top_k_indices = torch.topk(output, k=5, dim=-1)
    keywords = [bert_tokenizer.decode(token.item()) for token in top_k_indices.flatten()]
    concepts = output.mean(dim=0).tolist()
    summary = {
        "keywords": keywords,
        "concepts": concepts,
        "summary": " ".join(keywords),
    }
    return summary
```

```
def analyze_syllabus(text, bert_model, bert_tokenizer):
    blocks = text.split("\n\n")
    summary = [analyze_syllabus_block(block, bert_model, bert_tokenizer) for block in blocks]
    return summary

def print_summary(summary):
    for item in summary:
        print("**Keywords:**", ", ".join(item["keywords"]))
        print("**Concepts (mean of token embeddings):**", item["concepts"])
        print("**Summary:**", item["summary"])
        print("\n" + "="*40 + "\n")  # Add a separator between blocks

def load_bert_model_and_tokenizer():
    bert_model = BertModel.from_pretrained("bert-base-uncased")
    bert_tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
    return bert_model, bert_tokenizer

if __name__ == "__main__":
    pdf_file_path = "syllabus.pdf"

    try:
        bert_model, bert_tokenizer = load_bert_model_and_tokenizer()
```

```
        syllabus_text = extract_text_from_pdf(pdf_file_path)
        analysis_result   =   analyze_syllabus(syllabus_text,
bert_model, bert_tokenizer)
        print_summary(analysis_result)

    except Exception as e:
        print(f"Error: {e}")
```