Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Курсова робота

освітній ступінь – магістр

на тему: «**Пошук та визначення плагіату в текстах українською мовою**»

Виконав: студент 1-го року навчання освітньої програми «Прикладна математика», спеціальності 113 Прикладна математика

Величко Ростислав Анатолійович

Керівник Глибовець А.М.,_____

доктор технічних наук

Рецензент_____
(прізвище та ініціали)

Курсова робота захищена
з оцінкою _____

Секретар ЕК _____
«____» _____ 20____ р.

Київ – 2023

ЗМІСТ

ВСТУП

У сучасному світі, коли доступ до інформації став безпрецедентно швидким та легким, питання збереження інтелектуальної власності стає дедалі більш актуальним. Особливо важливою проблемою є виявлення плагіату, коли авторські текстові матеріали без дозволу використовуються іншими авторами. Це може стати причиною порушення авторських прав та завдати шкоду репутації автора оригінального матеріалу.

У даній курсовій роботі розглядається проблема виявлення плагіату українських текстів за допомогою семантичного підходу Латентного розміщення Дирихле (LDA) та Латенто-семантичного аналізу (LSA). Семантичні моделі дозволяють виявляти плагіат на основі аналізу значень та змісту тексту, а не лише на основі порівняння структури та форматування.

Метою роботи є розробка програми на мові Python, яка визначатиме рівень плагіату в українських текстах за допомогою семантичного моделювання (LDA та LSA).

Завдання роботи:

1. Провести аналіз літератури з питань тематичного моделювання, косинусної подібності та Евклідової відстані.
2. Розробити програму на мові Python, яка буде здійснювати підготовку тексту (очищення від зайвих символів, лематизація, токенізація) та застосовувати тематичні моделі LDA та LSA.
3. Реалізувати обчислення косинусної подібності та Евклідової відстані між текстами.
4. Розробити алгоритм визначення рівня плагіату, який буде враховувати результати застосування тематичних моделей та метрик подібності.

5. Провести експериментальне дослідження розробленої програми на корпусі українських текстів для оцінки її ефективності та точності.

6. Зробити висновки та рекомендації щодо можливих покращень програми та перспектив подальших досліджень в даній області.

Об'єктом дослідження є українські тексти різної тематики та жанрів, які мають потенційний ризик плагіату.

Методи дослідження — аналіз наукової літератури

Наукова новизна - Отримання результатів використання LDA, LSA, для визначення плагіату в українських текстах має наукову новизну, оскільки не було проведено аналізу такого роду на українському мовному корпусі. Це дослідження дозволить покращити ефективність виявлення плагіату в українських текстах та забезпечить нові можливості для використання машинного навчання в цій сфері.

Практичне значення отриманих результатів - Отримані результати мають практичне значення для виявлення плагіату в українських текстах. Розроблена програма, яка використовує LDA, LSA може бути використана для перевірки авторства текстів. Отримані результати можуть допомогти виявити можливі випадки плагіату та запобігти їх поширенню, зберігаючи таким чином інтелектуальну власність та зберігаючи високі стандарти етики та наукової доброчесності.

1.Види плагіату

Список найпоширеніших видів плагіату[1]:

Повний плагіат.

Повний плагіат може виникнути, якщо ви скопіюєте чужу роботу та подасте її як свою. Ця форма є ні чим іншим, як крадіжкою.

Прямий плагіат

Концепція повного плагіату дуже схожа на концепцію прямого плагіату, за винятком кількох речей. Прямий плагіат означає копіювання кожного слова розділу чиєїсь роботи. Якщо повний плагіат включає плагіат усього завдання, прямий плагіат стосується певних розділів або абзаців.

Самоплагіат

Самоплагіат також відомий як автоплагіат. Ви можете вчинити самоплагіат, якщо надсилаєте свою стару роботу або додаєте деякі її частини за відсутності дозволу від усіх залучених професорів. Хоча в більшості випадків самоплагіат не є незаконним, він може призвести до етичних проблем, оскільки є нечесним актом і навіть літературною крадіжкою.

Перефразування плагіату

Перефразування полягає у використанні ваших слів, щоб представити чужу ідею, не згадуючи її. Щоразу, коли ви представляєте ідею іншої людини, не цитуючи її, це стає крадіжкою її роботи. Зараз існують різні способи перефразувати фрагмент тексту, включаючи зміну

структури речення, додавання синонімів, зміну голосу тексту та багато інших.

Печворк-плагіат

Також відомий як плагіат Mosiac, плагіат Patchwork відноситься до взяття фраз, ідей і джерел з різних джерел і поєднання їх разом, щоб представити як новий текст.

Плагіат джерел

Плагіат на основі джерел може бути важко зрозуміти, оскільки йдеться про цитування. Це відбувається, коли автор правильно цитує джерела, але не представляє їх. Такі ситуації, як посилання на неправильні джерела, також підпадають під цей вид плагіату.

Випадковий плагіат

Як випливає з назви, випадковий плагіат – це те, що відбувається випадково чи ненавмисно. Це може включати невдалу спробу зрозуміти керівні принципи університету, забуття посилання на джерела або недодавання цитат навколо згаданого матеріалу.

2. Підходи до виявлення плагіату

Виявлення плагіату є однією з важливих задач у сфері обробки природньої мови (NLP). За допомогою NLP, можна виявляти та аналізувати структуру тексту, визначати стиль письма та інші характеристики, які можуть вказувати на наявність плагіату.

NLP використовується для розуміння та обробки природньої мови,

яка є складною для обробки комп'ютерами через її семантичну та контекстуальну складність. Алгоритми NLP допомагають аналізувати та обробляти текстові дані та забезпечують можливість виявлення плагіату.

Одним з популярних методів виявлення плагіату є використання тематичних моделей, таких як LDA та LSA, які базуються на аналізі вживаних слів та їх семантичних співвідношеннях. Ці моделі дозволяють виявити та порівняти структуру текстів, що забезпечує можливість виявлення плагіату. Далі ми розглянемо, як працюють дані моделі.

3. Моделі, які використовувались для визначення плагіату

3.1. Латентне розміщення Дирихле (LDA)

Латентне розміщення Дирихле (LDA) є методом тематичного моделювання, який дозволяє автоматично виявляти теми, які присутні в наборі текстів. LDA використовуєтсья для розкриття тем, які знаходяться під поверхнею тексту, і є інструментом для виявлення складних тематичних зв'язків в даних.

У контексті визначення плагіату в текстах, LDA може використовуватись для порівняння двох текстів на основі їх тематичного профілю. Алгоритм LDA перетворює кожен текст у вектор, що містить ймовірності його належності до кожної теми в моделі.

Ми можемо реалізувати LDA за допомою методів “торба слів” (bag-of-words) та TF-IDF (Term frequency-inverse document frequency)

3.1.1 “Торба слів” (bag-of-words)

“торба слів” (bag-of-words) — це техніка обробки природної мови

для моделювання тексту. Цей підхід є простим і гнучким способом вилучення функцій із документів.[2]

“торба слів” (bag-of-words) — це представлення тексту, що описує наявність слів у документі. Ми просто відстежуємо кількість слів і ігноруємо граматичні деталі та порядок слів. Його називають «торбою» слів, оскільки будь-яка інформація про порядок або структуру слів у документі відкидається. Модель стосується лише того, чи зустрічаються відомі слова в документі, а не того, де в документі.[2]

Розгялнемо для прикладу 2 речень[2]:

Речення 1: “Welcome to Great Learning, Now start learning ”

Речення 2: “Learning is a good practice ”.

Спочатку складається список усіх слів з 2 речень.

Далі кожному слові в реченні присвоюється число, скільки разів дане слово зустрічається в реченні, тобто частота.

Для речення 1 частоти будуть виглядати наступним чином[2]:

Word	Frequency
Welcome	1
to	1
Great	1
Learning	1
,	1
Now	1
start	1
learning	1
is	0
a	0
good	0
practice	0

Далі ми записуємо наші частоти в вектор, результатом є ось такий вектор[2]:

[1,1,1,1,1,1,1,1,0,0,0]

Відповідним чином для 2 речення отримали вектор[2]:

[0,0,0,0,0,0,0,1,1,1,1,1]

Але для ефективного використання потрібно спочатку зробити попередню обробку тексту, щоб вилучити знаки пунктуації, однакові слова з верхім та нижнім регістрами.

3.1.2. TF-IDF (Term frequency-inverse document frequency)

Частота термінів (TF) працює, дивлячись на частоту певного терміну, який вас цікавить, відносно документа. Існує кілька заходів або способів визначення частоти[3]:

- Кількість разів, коли слово з’являється в документі (необроблена кількість).

- Частота термінів, скоригована відповідно до довжини документа (необхідна кількість повторень, поділена на кількість слів у документі).

- Логарифмічно масштабована частота (наприклад, log(1 + кількість разів,коли слово з'являється в документі)).

- Логічна частота (наприклад, 1, якщо термін зустрічається, або 0, якщо термін не зустрічається в документі).

Зворотна частота документа (IDF) розглядає, наскільки поширеним (або незвичайним) слово є в корпусі. IDF обчислюється наступним чином(3):

$$idf\,(t, D) = log\,(\ \frac{N}{count\ (d \in D : t \in d)}\ )$$

де t — це термін (слово), загальність якого ми шукаємо, N — кількість документів (d) у корпусі (D). Знаменник — це просто кількість документів, у яких термін t з'являється.

Зворотна частота документа використовується для того, щоб збільшити вплив рідскісних термінів, мінізувавши вплив термінів, які часто зустрічаються.

Векторизація TF-IDF передбачає обчислення оцінки TF-IDF для

кожного слова у вашому корпусі відносно цього документа, а потім поміщення цієї інформації у вектор. Таким чином, кожен документ у вашому корпусі матиме власний вектор, і вектор матиме оцінку TF-IDF для кожного окремого слова у всій колекції документів[3].

Приклад використання TF-IDF для двох речень[3]:

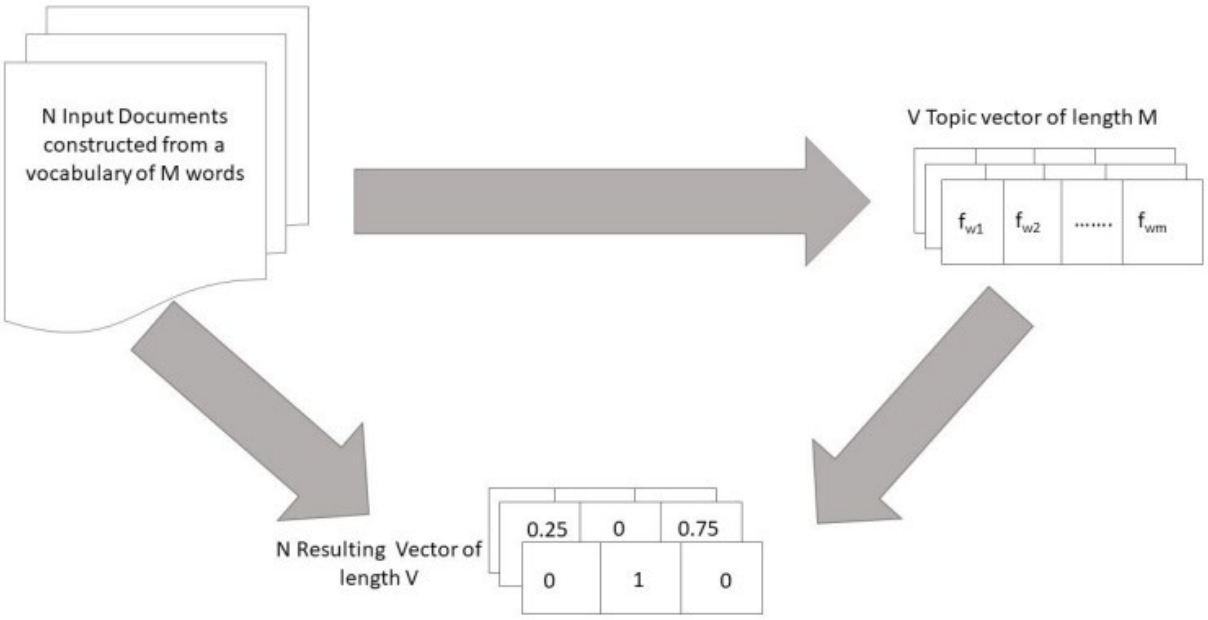
Речення A - “The car is driven on the road”,

Речення B - “The truck is driven on the highway”

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

3.1.3. Архітектура Латентного розміщення Дирихле (LDA)

На даному зображенні показана архітектура Латентного розміщення Дирихле[4]:



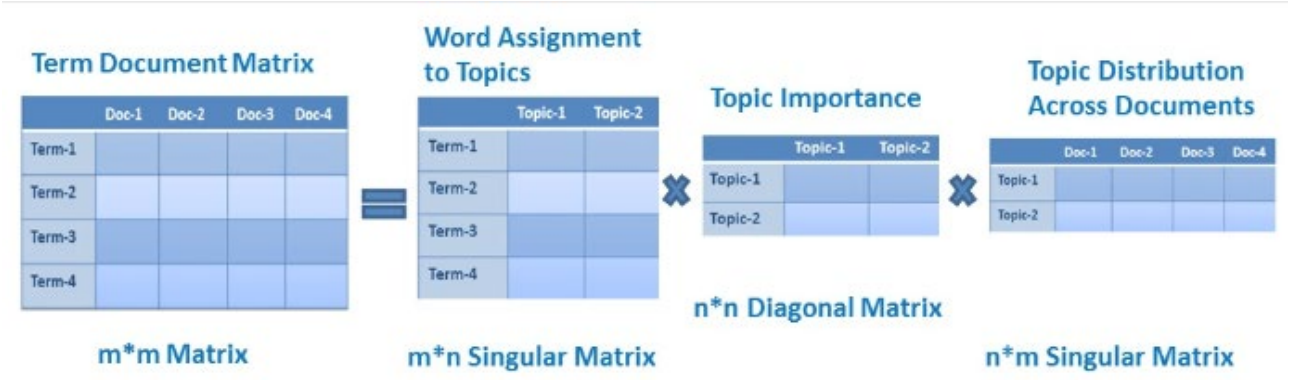
У LDA кожен документ представлений як “торба слів”, тобто вектор фіксованої довжини, яка дорівнює розміру словникового запасу. Кожен вимір цього вектора відповідає частоті появи слова в документі. Щоб підвищити продуктивність LDA деякі необхідна попередня обробка тексту.[4]

Кінцевим результатом тематичного моделювання, виконаного підходом LDA, є набір тем і супутні документи. Для кожного документа існує вектор довжини, що дорівнює кількості тем та містить ймовірність зв’язку з темою. У налаштуваннях моделі є можливість визначити кількість бажаних тем.[4]

3.2. Латенто-семантичний аналіз (LSA)

LSA (латентний семантичний аналіз). LSA використовує модель “торба слів” , яка призводить до матриці термін-документ (поява термінів

у документі). Рядки представляють терміни, а стовпці представляють документи. LSA вивчає приховані теми, виконуючи декомпозицію матриці на матриці термінів документа за допомогою декомпозиції сингулярного значення.[5]



3.2.1. Сингулярне розкладання (SVD)

Сингулярне розкладання (SVD) — це метод матричної факторизації, який представляє матрицю у вигляді добутку двох матриць.[5]

$$M=U\Sigma V^*$$

M є матрицею m×m

U — ліва сингулярна матриця m×n

Σ — діагональна матриця n×n із невід'ємними дійсними числами.

V є правою сингулярною матрицею m×n

V* — це матриця n×m, яка є транспонуванням V.

Ідентична матриця: це квадратна матриця, у якій усі елементи головної діагоналі є одиницями, а всі інші елементи — нулями.

Діагональна матриця: це матриця, у якій усі елементи, крім головної діагоналі, дорівнюють нулю.

Сингулярна матриця: матриця є сингулярною, якщо її визначник дорівнює 0, або квадратна матриця, яка не має зворотної матриці.

4. Косинусна подібність та Евклідова відстань

Косинусна подібність — це показник, який використовується для вимірювання подібності двох векторів. Зокрема, він вимірює подібність у напрямку або орієнтації векторів, ігноруючи відмінності в їх величині або масштабі.[6]

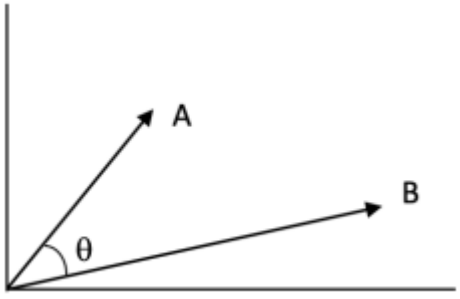
Ми визначаємо косинусну подібність математично як скалярний добуток векторів, поділений на їх величину. Наприклад, якщо ми маємо два вектори, A і B, подібність між ними обчислюється як[6]:

$$similarity(A, B) = cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

де:

– θ — кут між веторами

– A та B — вектори.



Значення косинусної подібності знаходиться в межах (-1;1)

Зазвичай, для використання в загальній сфері можна використовувати значення порогів, які є стандартними для більшості задач з виявлення плагіату. Наприклад, значення порогу cos_sim може бути встановлене на 0.8, щоб вважати документи схожими на достатньо

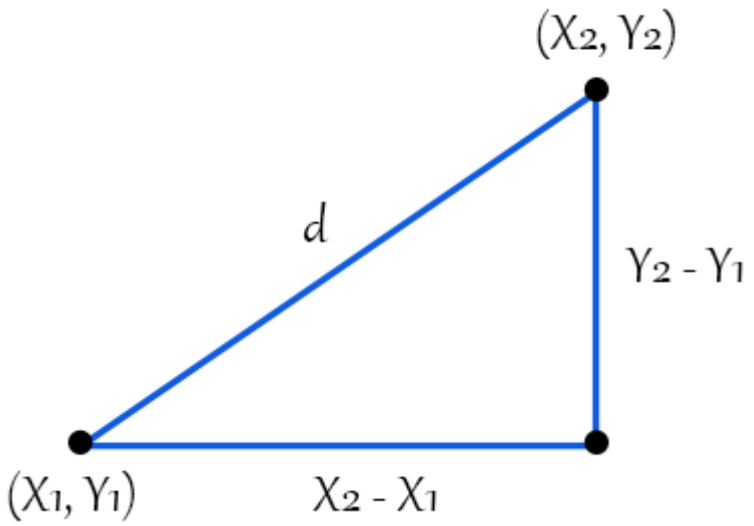
високому рівні.

Евклідова відстань обчислює відстань між двома дійсними векторами.

Ми будемо використовувати евклідову відстань під час обчислення відстані між двома рядками даних, які мають числові значення.[7]

Якщо стовпці мають значення з різними масштабами, прийнято нормалізувати чи стандартизувати числові значення в усіх стовпцях перед обчисленням евклідової відстані. Інакше стовпці з великими значеннями домінуватимуть у вимірюванні відстані.[7]

$$d = |\mathbf{x} - \mathbf{y}| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$



5. Визначення плагіату в українських текстах

5.1. Огляд реалізацій LDA та LSA

Моделі LDA та LSA можна навчити для довільних текстів. Навчання таких моделей потребує великого корпусу текстів, написаних українською мовою, та значну кількість ресурсів для їх обчислення та тренування. В даній роботі ми будемо використовувати дані моделі таким чином: обидві моделі LDA та LSA будуть навчатись на корпусі текстів, які будуть міститись в файлах, які ми будемо перевіряти на плагіат. Далі вже натреновані моделі будуть застосовуватись до кожного файлу окремо.

5.2. Програма для визначення плагіату українських текстів

Спочатку ми робимо попередню обробку тексту з метою підготовки його до подальшого його застосування. А саме:

– перетворюємо текст на рядок з малих літер

– видаляємо знаки пунктуації за допомогою функції *translate* та модуля *string.punctuation*

– розбиваємо текст на токени (окремі слова) за допомогою функції *word_tokenize* з модуля *nltk*

– видаляємо числа з токенів за допомогою фільтрації

```
# Preprocess the text
def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()

    # Remove punctuation
    text = text.translate(str.maketrans("", "", string.punctuation))

    # Tokenize text
    tokens = nltk.word_tokenize(text)
    tokens = [token for token in tokens if not token.isdigit()]

    return tokens
```

Далі ми створюємо корпус — список документів, кожен з яких представлений у векторній формі, де кожен компонент відповідає частоті

зустрічі слова в документі.

Функція отримує на вхід список текстів, які були попередньо очищені та токенізовані. Спочатку вона створює словник (*dictionary*) для всіх слів в текстах. Далі, вона створює корпус (*corpus*), використовуючи *dictionary* та *doc2bow* - функцію, яка перетворює документи в масиви токенів та їх кількість, щоб можна було виконати тематичне моделювання.

Функція повертає словник та корпус.

```
# Create the corpus
def create_corpus(texts):
    dictionary = corpora.Dictionary(texts)
    corpus = [dictionary.doc2bow(text) for text in texts]
    return dictionary, corpus
```

Потім ми за допомогою функції *apply_lda()*, яка використовує бібліотеку Gensim, щоб застосовуємо модель LDA (Latent Dirichlet Allocation) до корпусу документів і повертаємо результати моделювання. Та за допомогою функції *apply_lsa()*, яка також використовує Gensim, застосуємо модель LSA (Latent Semantic Analysis) до корпусу документів і повертаємо результати моделювання.

У обох функціях використовується парметр *num_topics*, який визначає кількість тем, які будуть знайдені моделями LDA та LSA. Даний параметр встановлений як 2, оскільки ми будемо розглядати короткі українські тексти, в яких міститься не більше 2 тем.

```
# Apply LDA
def apply_lda(corpus, num_topics=2):
    lda_model = models.LdaModel(corpus, num_topics=num_topics)
    return lda_model

# Apply LSA
def apply_lsa(corpus, num_topics=2):
    lsa_model = models.LsiModel(corpus, num_topics=num_topics)
    return lsa_model
```

Потім ми ініціалізуємо функцію *represent_document.* Дана функція приймає на вхід список токенів, словник, та моделі LDA та LSA. Функція перетворює список токенів у представлення “торби слів” (bag-of-words) за допомогою словника, використовуючи метод *doc2bow()*. Далі, вона

використовує натреновані моделі LDA та LSA для перетворення даних у векторну форму. Результатом є представлення документу в LDA та LSA просторі.

```
# Represent the document as a vector
def represent_document(tokens, dictionary, lda_model, lsa_model):
    # Flatten nested list of tokens
    tokens = list(itertools.chain.from_iterable(tokens))

    # Create bag-of-words representation
    text_bow = dictionary.doc2bow(tokens)

    # Create LDA representation
    lda_vector = lda_model[text_bow]

    # Create LSA representation
    lsa_vector = lsa_model[text_bow]

    return lda_vector, lsa_vector
```

Далі ми реалізуємо функції для визначення косинусної подібності та евклідової відстані.

```
# Compare the documents using cosine similarity
def compare_documents_cosine(doc1, doc2):
    doc1 = np.asarray(doc1)
    doc2 = np.asarray(doc2)
    if doc1.shape != doc2.shape:
        min_shape = min(doc1.shape[0], doc2.shape[0])
        doc1 = doc1[:min_shape]
        doc2 = doc2[:min_shape]
    sim = cosine_similarity(doc1.reshape(1, -1), doc2.reshape(1, -1))[0][0]
    return sim

# Compare the documents using Euclidean distance
def compare_documents_euclidean(doc1, doc2):
    doc1 = np.asarray(doc1)
    doc2 = np.asarray(doc2)

    # Normalize the document vectors
    doc1_norm = doc1 / np.linalg.norm(doc1)
    doc2_norm = doc2 / np.linalg.norm(doc2)

    # Compute the Euclidean distance between normalized vectors
    sim = euclidean_distances(doc1_norm.reshape(1, -1), doc2_norm.reshape(1, -1))[0][0]
    return sim
```

В іншій частині коду ми:

– завантажуємо 2 файли, які будемо перевіряти між собою на плагіат

– розбиваємо кожен файл на токени та робимо попередню обробку тексту

– об'єднуємо всі токени з обох файлів в один список

– створюємо словник та корпус з усіх токенів з файлів

– застосовуємо моделі LDA та LSA до корпусу

– представляємо кожен файл як вектори LDA та LSA

– обчислюємо косинусну подібність та Евклідову відстань між веторами LDA та LSA до кожного файлу

– визначаємо, чи можуть файли містити плагіат, порівнюючи косинусну подібність та Евклідову відстань за певними пороговими значеннями.

```
# Load the files and preprocess the texts
with open("file2.txt", "r", encoding="utf-8") as f1:
    file1_text = f1.read()
file1_sentences = nltk.sent_tokenize(file1_text)
file1_tokens = [preprocess_text(sentence) for sentence in file1_sentences]

with open("file3.txt", "r", encoding="utf-8") as f2:
    file2_text = f2.read()
file2_sentences = nltk.sent_tokenize(file2_text)
file2_tokens = [preprocess_text(sentence) for sentence in file2_sentences]

# Create the corpus and models
all_tokens = file1_tokens + file2_tokens
dictionary, corpus = create_corpus(all_tokens)
lda_model = apply_lda(corpus)
lsa_model = apply_lsa(corpus)

# Represent the documents as vectors
file1_lda, file1_lsa = represent_document(file1_tokens, dictionary, lda_model, lsa_model)
file2_lda, file2_lsa = represent_document(file2_tokens, dictionary, lda_model, lsa_model)

# Compare the documents using cosine similarity
cos_sim_lda = compare_documents_cosine(file1_lda, file2_lda)
cos_sim_lsa = compare_documents_cosine(file1_lsa, file2_lsa)
print("Cosine Similarity LDA:", cos_sim_lda)
print("Cosine Similarity LSA:", cos_sim_lsa)

# Compare the documents using Euclidean distance
euclidean_dist_lsa = compare_documents_euclidean(file1_lsa, file2_lsa)
euclidean_dist_lda = compare_documents_euclidean(file1_lda, file2_lda)
print("Euclidean Distance LDA:", euclidean_dist_lda)
print("Euclidean Distance LSA:", euclidean_dist_lsa)

# Check for plagiarism using LDA and LSA
if cos_sim_lda > 0.8 and euclidean_dist_lda < 0.5:
    print("LDA: The documents may contain plagiarized content.")
else:
    print("LDA: The documents do not appear to contain plagiarized content.")

if cos_sim_lsa > 0.8 and euclidean_dist_lsa < 0.5:
    print("LSA: The documents may contain plagiarized content.")
else:
    print("LSA: The documents do not appear to contain plagiarized content.")
```

Значення порогу *cos_sim* було встановлене на 0.8, щоб вважати документи схожими на достатньо високому рівні.

Значення порогу *euclidean_dist* було втсановлено на 0.5, оскільки в загальному випадку використовуються значення в межах від 0.3 до 0.7.

В результаті програма виводить значення косинусної подібності та Евклідової відстані для двох моделей: LDA та LSA. На основі порогових значенням виводить повідомлення про те, чи міститься потенційно плагіат в даних файлах, чи ні.

Тестування програми було виконане для декількох випадків українських текстів.

Перший випадок: українські тексти на різні теми та потенційно не містять плагіат.

Документ 1:

Природа (від лат. natura) – це, перш за все, ми самі, а також все існуюче, весь світ в об'єктивній різноманітності його проявів.
Поняття «природа» охоплює все суще, весь Всесвіт, воно близьке до поняття «матерія». Вона не має ні початку, ні кінця, вона безкінечна у просторі й часі, перебуває у безперервному русі, змінах.
Найбільш поширеним є розуміння природи як сукупності об'єктивних умов існування людства, його навколишнього середовища.
Природа – це весь навколишній світ:
рослини, тварини, ліси, моря, гори, рівнини та й сама людина…

Документ 2:

Python (укр. Пайтон) — високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність.

Вступ

Мова Python: що це?
Можливості мови програмування Python
Що таке мова Python: особливості застосування
Бібліотеки Python
Найвідоміші фреймворки для мови програмування Python
Популярні Python IDE
Мова Python: що це?
Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи

Результат роботи програми:

```
Cosine Similarity LDA: 0.8021929040061749
Cosine Similarity LSA: 0.16585473535840123
Euclidean Distance LDA: 0.628978689613289
Euclidean Distance LSA: 1.2916232149056464
LDA: The documents do not appear to contain plagiarized content.
LSA: The documents do not appear to contain plagiarized content.
```

Як і очікувалось результатом роботи програми є повідомлення про те, що не знайдено потенційного плагіату між цими документами для обох моделей LDA та LSA.

Другий випадок: українські тексти на однакові теми, та потенційно містять плагіат. Документ 1:

Що таке Python?
Особливості та переваги використання мови програмування Python

PythonPython проста у використанні, та водночас повноцінна мова програмування, що надає набагато більше засобів для структурування і підтримки великих програм, ніж shell. З іншого боку, вона краще за C обробляє помилки, і, будучи мовою дуже високого рівня, має вбудовані типи даних високого рівня, такі як гнучкі масиви і словники, ефективна реалізація яких на C потребує значних витрат часу.

Завдяки більш загальним типам даних, Python застосовують до більш широкого кола задач, ніж Awk і навіть Perl, у той ж час багато речей на мові Python робляться настільки ж просто.

Документ 2:

Пайтон (Python) — це потужна мова програмування, якою легко оволодіти. Вона має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Пайтона, динамічна обробка типів, а також те, що це інтерпретована мова, роблять його ідеальним для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови Пайтон і багата стандартна бібліотека (як код-джерело, так і бінарні дистрибутиви для усіх головних операційних систем) можуть бути отримані з сайту Пайтона, і можуть вільно розповсюджуватися.

Результат роботи програми:

```
Cosine Similarity LDA: 0.9385624272992783
Cosine Similarity LSA: 0.9859513798546599
Euclidean Distance LDA: 0.3505355123257031
Euclidean Distance LSA: 0.16762231441750247
LDA: The documents may contain plagiarized content.
LSA: The documents may contain plagiarized content.
```

Результатом робот програми є повідомлення про те, що потенційно документи 1 та 2 є плагіатом.

В ході тестування були також випадки коли моделі LDA та LSA давали різні результати про те, чи є дані документи плагіатом, чи ні. Модель LSA частіше давала правильні результати.

ВИСНОВКИ

У результаті розробки програми для визначення плагіату українських текстів за допомогою моделей LDA та LSA, було успішно реалізовано функціонал, що дозволяє завантажувати два тексти, проводити їх попередню обробку, створювати корпус, застосовувати моделі LDA та LSA, щоб представити тексти у вигляді векторів та порівнювати їх за допомогою метрик косинусної схожості та Евклідової відстані.

Використання моделі LDA дозволяє розпізнавати схожість між текстами з точки зору використання подібних тем, що може бути корисно при визначенні схожості між тексти, які містять багато спільних тематичних кластерів. Модель LSA ж, в свою чергу, може бути корисною при визначенні схожості між текстами, які містять схожі слова та використовують подібні словосполучення.

Результати, отримані за допомогою реалізованої програми, можуть бути використані як один з методів визначення плагіату українських текстів, але не є єдиним і не можуть бути використані як остаточний доказ плагіату.

Програму можна вдосконалити, а саме тренувати моделі LDA та LSA на великих корпусах даних, використовувати дані моделі в комбінації з іншими відомими, наприклад моделями нейронних мереж, розробити інтерфейс для зручного використання іншими користувачами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Види плагіату / Смодін [Електронний ресурс] - https://smodin.io/uk/blog/types-of-plagiarism/
2. An Introduction to Bag of Words (BoW) | What is Bag of Words? / Great Learning Team [Електронний ресурс] - https://www.mygreatlearning.com/blog/bag-of-words/
3. Understanding TF-IDF for Machine Learning / Anirudha Simha [Електронний ресурс] - https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/
4. SPUFCL (Scientific Publication Classifier): A Human-Readable Labelling System for Scientific Publications / Noemi Scarpato, Alessandra Pieroni, Michela Montorsi – 2021 - https://www.mdpi.com/2076-3417/11/19/9154
5. Latent Semantic Analysis using Python / Avinash Navlani [Електронний ресурс] - https://www.datacamp.com/tutorial/discovering-hidden-topics-python
6. Cosine similarity / Fatih Karabiber [Електронний ресурс] - https://www.learndatasci.com/glossary/cosine-similarity/
7. 4 Distance Measures for Machine Learning / Jason Brownlee [Електронний ресурс] - https://machinelearningmastery.com/distance-measures-for-machine-learning/