НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
Факультет Інформатики та обчислювальної техніки
Кафедра Інформаційних систем та технологій

ЛАБОРАТОРНА РОБОТА №5

з дисципліни «Обробка та аналіз текстових даних на Python»

на тему «Тематичне моделювання з використанням LDA»

Варіант №2

Виконала: студентка групи ІС-з21
Коломієць Катерина Миколаївна
05.06.2025

Перевірив: асист. Мягкий М. Ю.

Київ – 2025

GitHub: https://github.com/kate-miiets-uni/oatd-py

Теоретичні відомості

Алгоритм Latent Dirichlet Allocation (LDA), який є статистичною моделлю для виявлення "прихованих" (латентних) тем у колекції документів. Це один з найпопулярніших методів тематичного моделювання.

LDA базується на двох ключових припущеннях:

1) кожен документ є сумішшю декількох тем. Наприклад, стаття про економіку може мати 80% теми "Фінанси" і 20% теми "Державна політика";

2) кожна тема є сумішшю декількох слів. Наприклад, тема "Фінанси" може мати високу ймовірність для слів "банк", "інвестиції", "акції", а тема "Державна політика" – для слів "уряд", "закон", "реформа".

Завдання LDA – "розкласти" корпус документів на ці приховані компоненти:

- які теми існують у корпусі?
- які слова найбільше характеризують кожну тему?
- який розподіл тем має кожен документ?

Усі висновки LDA базуються на ймовірностях. Слово не "належить" темі абсолютно, а має певну ймовірність належати до неї. Документ не "є" темою, а є "сумішшю" тем з певними ймовірностями.

LdaModel – це потужний інструмент, який, використовуючи статистичні ймовірності, допомагає виявити приховану тематичну структуру у великих колекціях текстових даних, роблячи їх більш зрозумілими та доступними для аналізу.

Хід роботи

1. Імпортуємо необхідні бібліотеки. У цій роботі використовуємо:

- requests – для імпортування html-документу новини з сайту;
- bs4.BeautifulSoup – для очищення тексту від html тегів;
- re – для використання регулярних виразів для очищення текстів;
- gensim – для неконтрольованого тематичного моделювання.

```
import requests
import re
from bs4 import BeautifulSoup
from gensim.corpora import Dictionary
from gensim.models import LdaModel
```

2. Збираємо 5 новин про економіку в список для легшого їхнього імпортування. Створюємо список для документів (текстів), які будемо опрацьовувати.

```
urls = [
    'https://www.bbc.com/ukrainian/articles/cn7zr18rx46o',
    'https://www.bbc.com/ukrainian/articles/cjwvx7783xvo',
    'https://www.bbc.com/ukrainian/articles/czd30mqlv2lo',
    'https://www.bbc.com/ukrainian/articles/c0r55z299ggo',
    'https://www.bbc.com/ukrainian/articles/cvgqld6mld9o'
]

# Список для зберігання документів з текстами новин
documents = []
```

3. Імпортуємо усі тексти новин за допомогою циклу, у якому для кожної новини надсилаємо запит до сервера про отримання html-документа новини, перевіряємо чи запит вдалий. Якщо так, видаляємо за допомогою BeautifulSoup усі теги та додаємо очищений текст в список документів.

```
# Для кожного посилання зі списку посилань
for url in urls:
    # Надсилаємо запит на сервер про отримання тексту новини
    response = requests.get(url)
    # Якщо запит вдалий
    if response.status_code == 200:
        # Видаляємо з текста новини усі теги
        soup = BeautifulSoup(response.text, features: 'html.parser')
        # Видобуваємо текст з тегів <p>
        text = ' '.join([p.text for p in soup.find_all('p')])  # Отримуємо текст із тегів <p>
        # Додаємо текст у список з документами
        documents.append(text)
```

Отримаємо наступного вигляду вміст статей:

```
Автор фото, BBC/Getty Images Угода про ресурси, яку Дональд Трамп уклав з Україною, не прив
```

4. Очищуємо текст від стоп-слів та коротких слів (таких, що містять менше 3-х символів). Також приводимо всі слова до нижнього регістру для уніфікації слів. Видаляємо всі символи окрім літер, цифр та _. Повертаємо список слів, а не рядок, адже надалі словник gensim.corpora.Dictionary буде очікувати список токенів кожної статті.

```python
# Функція приводить текст до нижнього регістру, очищає його від пунктуації та с
def clean_text(text):  1 usage
    # Сет стоп-слів
    stop_words = {"щоб", "про", "але", "для", "від", "через", "також", "якщо",
    # Приводимо до нижнього регістру для уніфікації слів
    text = text.lower()
    # Видаляємо усі символи окрім літер, цифр та _
    text = re.sub( pattern: r'\W+',  repl: ' ', text)
    # Розділяємо текст на слова для видалення стоп-слів
    words = text.split()
    # Видаляємо стоп-слова та короткі слова (менше 3-х символів)
    words = [word for word in words if word not in stop_words and len(word) > 2
    return words  # Повертаємо список слів, а не об'єднаний рядок
```

```python
cleaned_documents = [clean_text(doc) for doc in documents]
print(cleaned_documents[0][:100])
```

Отримуємо наступного вигляду документи зі статтями:

```
['автор', 'фото', 'bbc', 'getty', 'images', 'угода', 'ресурси', 'яку', 'дональд', 'трамп',
```

5. Створюємо словник з очищених документів для слугування "перекладачем" між людським текстом і числовим форматом, який розуміють алгоритми, він призначає унікальний числовий ID кожному унікальному слову (токену). Створюємо корпус документів у форматі мішка слів (Bag of Words), тобто тепер кожен документ представлятиметься не як послідовність слів, а як набір пар (ідентифікатор_слова, кількість_входжень).

```python
# Створення словника (Dictionary) з очищених документів
dictionary = Dictionary(cleaned_documents)
print(f"\nКількість унікальних токенів у словнику: {len(dictionary)}")

# Створення корпусу документів у форматі BoW (Bag of Words)
corpus = [dictionary.doc2bow(doc) for doc in cleaned_documents]
print("\nПриклад елемента корпусу (BoW для першого документа):")
print(corpus[0][:10]) # Виводимо перші 10 пар (id слова, частота) для першого доку
```

Маємо приклад елемента корпусу:

```
Приклад елемента корпусу (BoW для першого документа):
[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 5), (6, 1), (7, 1), (8, 1), (9, 1)]
```

6. Задаємо кількість тем у нашій LDA-моделі та будуємо її. LdaModel приймає такі параметри: corpus=corpus - вхідні дані для LDA-моделі; num_topics=num_topics (де num_topics = 4) - параметр визначає кількість тем, які очкуємо знайти корпусі документів; id2word=dictionary - параметр надає моделі словник, який відображає числовий

ідентифікатор слова назад у його текстове представлення; random_state=100 - параметр встановлює початкове зерно для генератора випадкових чисел, встановлення фіксованого random_state забезпечить отримання однакових тем при кожному запуску програми; passes=10 - параметр вказує кількість проходів (епох) по всьому корпусу під час навчання; alpha='auto' - дозволяє автоматично визначити оптимальне значення alpha (гіперпараметр, що відповідає за розрідженість (sparsity) розподілу тем у документах) на основі даних під час навчання моделі.

```
# Побудова LDA-моделі
num_topics = 4
lda_model = LdaModel(corpus=corpus,
                     num_topics=num_topics,
                     id2word=dictionary,
                     random_state=100, # Для відтворюваності результатів
                     passes=10,
                     alpha='auto') # Автоматичне визначення параметрів alpha

# Виведення отриманих тем та їх ключових слів
print(f"\nLDA-модель з {num_topics} темами:")
for idx, topic in lda_model.print_topics(-1):
    print(f"Тема {idx}: {topic}")
```

Результат:

```
LDA-модель з 4 темами:
Тема 0: 0.019*"євро" + 0.013*"долара" + 0.009*"сша" + 0.008*"долар" + 0.006*"доларів" + 0.006*"курс" + 0.005*"валюти" + 0.004*"україни"
Тема 1: 0.010*"китаю" + 0.008*"сша" + 0.008*"китай" + 0.006*"трампа" + 0.005*"року" + 0.005*"автор" + 0.005*"фото" + 0.004*"років" + 0.
Тема 2: 0.013*"яйця" + 0.012*"ціни" + 0.010*"яєць" + 0.008*"сша" + 0.008*"року" + 0.004*"цього" + 0.004*"президент" + 0.004*"зростання"
Тема 3: 0.009*"україни" + 0.007*"сша" + 0.006*"україна" + 0.005*"ресурси" + 0.005*"bbc" + 0.005*"україні" + 0.005*"фонд" + 0.005*"фонду
```

Висновок

Отже, у ході лабораторної роботи ми дізналися про такий метод тематичного моделювання як LDA алгоритм, який допомогає виявити приховані теми у документах. Вперше дістали текст декількох статей відразу, очистили їх усі від зайвих символів, стоп-слів та коротких слів. Створили словник зі статей, а потім корпус з отриманого словника у форматі мішка слів. Побудували LDA-модель з 4-х тем та вивели їх на консоль.

Контрольні запитання

1. Як працює алгоритм LDA?

Лінійний дискримінантний аналіз (Linear Discriminant Analysis або LDA) — алгоритм класифікації та зниження розмірності, що дозволяє здійснювати розподіл класів найкращим чином. Основна ідея LDA полягає у припущенні про багатовимірний

нормальний розподіл ознак всередині класів та пошуку їхнього лінійного перетворення, яке максимізує міжкласову дисперсію та мінімізує внутрішньокласову. Іншими словами, об'єкти різних класів повинні мати нормальний розподіл і розташовуватися якомога далі один від одного, а об'єкти одного класу — якомога ближче.

Простіше кажучи, можна уявити, що у нас є багато документів, і необхідно зрозуміти, про що кожен документ. LDA намагається автоматично знайти набори слів, які часто зустрічаються разом у різних документах (це і є "теми"). Потім для кожного документа LDA визначає, які з цих тем найбільш ймовірно присутні в ньому. Щоб використовувати LDA, потрібно визначити кількість тем, які очікуються в корпусі. Це є одним із гіперпараметрів моделі.

2. Що означають слова в отриманих темах?

Слова в отриманих темах після застосування тематичного моделювання відображають характерні терміни, які часто зустрічаються в документах, що належать до цієї теми.

3. Як вибрати оптимальну кількість тем у моделі?

Для вибору оптимальної кількості тем використовують метрики перплексії (чим нижча перплексія, тим модель краще узагальнює дані і менш невпевнена у своїх прогнозах) та когерентності тем (чим вища когерентність, тим слова в темі частіше зустрічаються разом і тема є більш інтерпретованою та змістовною для людини). Візуалізація тем також допомагає оцінити їхню змістовність. Іноді найкращий вибір — це компроміс між метриками та інтерпретованістю.

4. Що таке токенізація та як вона використовується в LDA?

Токенізація — це процес розбиття тексту на менші одиниці, які називаються токенами. Зазвичай це окремі слова, але можуть бути й інші одиниці, такі як символи або підслова.

LDA працює з наборами слів кожного документа. Щоб отримати ці набори слів, сирий текст кожного документа спочатку токенізується. Після токенізації всі унікальні токени (слова) з усього корпусу документів утворюють словник. Кожен документ потім представляється як мультимножина (або вектор частот) своїх токенів зі словника. Порядок слів при цьому ігнорується — звідси й назва "мішок слів". LDA приймає на вхід ці представлення документів у вигляді частот токенів. Алгоритм потім намагається виявити приховані теми, аналізуючи, які набори слів часто зустрічаються разом у різних документах.

5. Як оцінити якість моделі?

Оцінити якість отриманої моделі можна за допомогою метрик або якісно шляхом аналізу змістовності та релевантності отриманих тем через перегляд топ-слів кожної теми

та оцінки того, наскільки легко їх інтерпретувати пересічному користувачеві, через перегляд документів кожної теми, через порівняння тем та обговорення результатів.

6. Які метрики використовують для оцінки LDA?

Для оцінки якості моделі тематичного моделювання, такої як LDA зазвичай використовують перплексію та когерентність тем.: перплексія - це міра того, наскільки добре модель прогнозує нові дані. Чим нижча перплексія, тим кращою вважається модель з точки зору її здатності до узагальнення. Низька перплексія означає, що модель більш впевнена у своїх прогнозах щодо ймовірності появи слів. Когерентність тем – це метрика, яка вимірює, наскільки семантично пов'язані слова всередині кожної виявленої теми. Чим вища когерентність тем, тим більш інтерпретованими та змістовними для людини є отримані теми. Висока когерентність означає, що слова в темі часто зустрічаються разом у документах.

7. Що таке когерентність у LDA?

Когерентність тем – це метрика, яка вимірює, наскільки семантично пов'язані слова всередині кожної виявленої теми. Чим вища когерентність тем, тим більш інтерпретованими та змістовними для людини є отримані теми. Висока когерентність означає, що слова в темі часто зустрічаються разом у документах.

Когерентність теми намагається відповісти на питання: "Наскільки ці слова в межах однієї теми насправді пов'язані між собою за змістом у реальному світі (або в контексті нашого корпусу документів)?

8. Як очистити текст перед тематичним аналізом?

Основні кроки очищення тексту включають переведення до нижнього регістру, видалення пунктуації, стоп-слів, чисел та спеціальних символів. Також застосовують стеммінг або лематизацію для зведення слів до їхньої основи. Вибір методів очищення залежить від конкретного завдання та даних.

9. Які бібліотеки використовуються для тематичного аналізу?

Gensim - це потужна бібліотека, спеціально розроблена для тематичного моделювання та аналізу подібності документів. Вона надає ефективні реалізації алгоритмів, таких як LDA (Latent Dirichlet Allocation), LSI (Latent Semantic Indexing) та HDP (Hierarchical Dirichlet Process). Gensim також має зручні інструменти для попередньої обробки тексту та візуалізації результатів.

10. Як інтерпретувати результати LDA?

Інтерпретація результатів LDA — це процес розуміння того, який зміст стоїть за кожною виявленою темою (на основі слів, що її складають) і як ці теми представлені в окремих документах вашого корпусу.

По закінченню навчання моделі LDA, на вихід отримуємо розподіл слів для кожної теми та розподіл тем для кожного документа. Щоб інтерпретувати їх, необхідно проаналізувати топ-слова на спільну ідею та чи всі вони підпадають під ту ідею (Наприклад, якщо для однієї теми топ-словами є "лікар", "пацієнт", "лікування", "здоров'я", можна інтерпретувати цю тему як "медицина"), переглянути вагомість розподілу тем на документ, а також зручним способом є візуалізація результатів.