

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА
АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

**РОЗРОБКА РЕКОМЕНДАЦІЙНОЇ НАВЧАЛЬНОЇ СИСТЕМИ НА
ОСНОВІ ОНТОЛОГІЙ**

**Текстова частина до курсової роботи
за спеціальністю „Інженерія програмного забезпечення” 121**

Керівник курсової роботи
к-т фіз.-мат.наук, доцент
Жежерун О.П.
“ ____ ” _____ 2020 р.

Виконав студент ФІ-3
Ніверовський М.М.
“ ____ ” _____ 2020 р.

Київ 2020

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА
АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики
ЗАТВЕРДЖУЮ

Викладач кафедри мультимедійних систем
к-т фіз.-мат.наук, доцент _____ Жежерун О.П
(підпис)

«___» _____ 2019р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу
студенту Ніверовському Микиті Миколайовичу
Факультету інформатики 3 р.н. бакалаврської програми
ТЕМА: Розробка рекомендаційної навчальної системи
на основі онтологій

Зміст текстової частини до курсової роботи:

Індивідуальне завдання

Вступ

Огляд теоретичного матеріалу і здійснення дослідження

Висновки

Список літератури

Додатки (за необхідністю)

Дата видачі «___» _____ 2019 р.

Керівник _____
(підпис)

Завдання отримав _____
(підпис)

**Тема: Розробка рекомендаційної навчальної системи на основі
онтологій**

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	15.10.2019	
2.	Огляд літератури за темою роботи	21.01.2020	
3.	Написання першого розділу	22.02.2020	
4.	Написання другого розділу	10.03.2020	
5.	Написання практичної частини	1.04.2020	
7.	Написання третього розділу	25.04.2020	
8.	Написання висновків та корегування роботи	30.04.2020	
9.	Створення презентації та написання доповіді.	8.05.2020	
10.	Здача курсової	10.05.2020	

ЗМІСТ

<i>Анотація</i>	5
<i>Використані скорочення</i>	6
<i>ВСТУП</i>	7
<i>РОЗДІЛ 1. ВИНИКНЕННЯ ТА ВИКОРИСТАННЯ ЛСА</i>	9
1.1 Загальна характеристика	9
1.2 Хронологія ЛСА	9
1.3 Основні застосування ЛСА	10
1.4 Переваги ЛСА	11
1.5 Недоліки ЛСА	11
<i>РОЗДІЛ 2. ОПИС РОБОТИ ЛСА НА ПРИКЛАДІ КЛАСИФІКАЦІЇ СТАТЕЙ</i>	12
2.1 Підготування даних	12
2.2 Побудова матриці вживаності	12
2.3 Розклад матриці	13
2.4 Обробка результатів	15
<i>РОЗДІЛ 3. РЕАЛІЗАЦІЯ ЛАТЕНТНО-СЕМАНТИЧНОГО АНАЛІЗУ</i> ..	16
3.1 Огляд засобів програмної розробки	16
3.2 Опис реалізації розробки	17
<i>ВИСНОВОК</i>	23
<i>СПИСОК ЛІТЕРАТУРИ</i>	24
<i>Додатки</i>	26
Додаток А	26
Додаток Б	27
Додаток В	28
Додаток Г	29
Додаток Ґ	30
Додаток Д	31

Анотація

Під час виконання даної роботи було досліджено техніку природного розпізнавання мови Латентно-семантичний аналіз. Було розглянуто хронологію досліджень на цю тему, основні застосування. Також було розібрано ЛСА на прикладі класифікації статей та реалізація цього приклада на мові програмування Python. Крім того, описується компоненти, які було використано для реалізації проекту.

Використані скорочення

ЛСА – латентно-семантичний аналіз (від англ. LSA – latent semantic analysis).

ЛСІ – латентно-семантичне індексування, альтернативна назва ЛСА, коли йде мова про пошук інформації.

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.

SVD (від англ. Singular-value decomposition) – сингулярний розклад матриці.

NLP (від англ. Natural Language Processing) – обробка природньої мови.

ВСТУП

Постановка задачі.

Дослідити техніку природного розпізнавання мови Латентно-семантичний аналіз та реалізувати один із засобів на мові програмування Python.

Актуальність дослідження.

Дослідження How much information[17], яке провели у 2009 році показало, що кількість інформації, яку споживає людина з 1986 року зросло у 5 разів. Навіть з'явився такий термін Інформаційний вибух.

Сучасна людина за день аналізує просто неймовірну кількість інформації.

Кожен день більшість людей читають різні статті, новини, документи тощо. Хтось їх пише. Для того, щоб якось шукати це в інтернеті, класифікувати, використовуються засоби розпізнавання природної мови.

Обробка природної мови зараз використовується майже у всіх галузях. А що ж ховається за словами обробка природної мови?

NLP – це розділ машинного навчання, який дозволяє комп'ютеру аналізувати природню мову людини.

Латентна семантичний аналіз – це один із засобів NLP, про який і буде мова у цій курсовій роботі.

Структура роботи.

Робота складається зі вступу, трьох розділів, списку літератури та доповнень.

У першому розділі дається загальний огляд тематики: хронологія досліджень ЛСА, переваги та недоліки, використання методів ЛСА.

У другому розділі обрана певна задача, а саме класифікація статей. Та описано алгоритм, як працює даний метод на прикладі.

У третьому розділі подана реалізація алгоритму на мові програмування Python: опис використаних бібліотек, засобів та середовищ розробки тощо.

РОЗДІЛ 1. ВИНИКНЕННЯ ТА ВИКОРИСТАННЯ ЛСА

1.1 Загальна характеристика

ЛСА – це метод обробки інформації, який аналізує набір документів та знаходить терміни, які там зустрічаються, та на основі цього виявляє характерні фактори, тематики, які характеризують зміст документу.

Визначають такі типи кореляції:

«слово-слово»;

«слово-параграф»;

«параграф-параграф».

Цими трьома типами мислить людина, співставляючи частини тексту зі змістом. Технологія ЛСА враховує не просто частотність вживання тексту, а й латенті(глибинні) зв'язки.

Проте ЛСА не враховує послідовність слів у реченні, тобто не враховує логіку та морфологію.

1.2 Хронологія ЛСА

Перша стаття по автоматичній класифікації документів “Automatic Document Classification”[1] була опублікована в журналі “Journal of the ACM” на початку 1963 року, у ній був уперше описан метод факторного аналізу як засіб для пошуку інформації. Факторний аналіз – це метод, який визначає зв'язок між значеннями змінних.

Алгоритм комп'ютерного пошуку інформації який використовував латентно-семантичну структуру був запатентований у 1988 командою дослідників: Deerwester; Scott C. (Chicago, IL), Dumais; Susan T. (Berkeley Heights, NJ), Furnas; George W. (Madison, NJ), Harshman; Richard A. (London, CA), Landauer; Thomas K. (Summit, NJ), Lochbaum; Karen E. (Chatham, NJ), Streeter; Lynn A. (Summit, NJ)[2].

У 1992 році метод латентно-семантичної індексації був уперше використан на конференції Hypertext'91 для розподілу рукописів між рецензентами. У ході опитування рецензентів, вони повідомили, що рукописи були досить добре розподілені, але це все одно було гірше, ніж коли це робить людина-експерт.[3]

У 1994 році було запатентовано методику комп'ютерного пошуку інформації серед декількох мов з використанням ЛСІ. Це було досліджено Thomas K. Landauer, Michael L. Littman.[4]

У журналі “Behavior Research Methods” у 1996 році була опублікована стаття “Latent semantic analysis for text-based research” [5], де PETERW. FOLTZ описав три експерименти, як можна використовувати ЛСА:

- 1) Співпадіння висновку та есе, тобто чи співпадає тема, яка піднімалась у есе до теми у висновку.
- 2) Оцінювання якості есе.
- 3) Вимірювання узгодженості та зрозумілості текстів.

Вперше детально описано алгоритм LSA був у 1999 році у роботі “An Introduction to Latent Semantic Analysis” Thomas K Landauer, Peter W. Foltz, Darrell Laham і потім поглиблено вивчено Scott Deerwester, Susan Dumais, George Furnas.

1.3 Основні застосування ЛСА

Близькі по значенню

Цей тип дозволяє подати у початкових параметрах термін або короткий текст та отримати список термінів, найближчих до поданих (у семантичному просторі LSA).

Порівняння у матриці

У цьому методі у вихідних параметрах подається n слів або текстів (параграфів), після обчислень виходить матриця $n \times n$, де показується, як кожне слово залежить від іншого.

Один-до-багатьох порівняння

За допомогою цього метода можна дізнатися, як відноситься одне слово або текст до n слів або текстів.

Багатомовний пошук

Цей тип дозволяє у вихідних параметрах подати запит однією мовою, а відповідь отримати іншими.

Оцінка есе

За допомогою цього методу можна оцінити якість написаного тексту.

1.4 Переваги ЛСА

- Метод вважається одним з кращих для пошуку глибинних (латентних) залежностей серед великої кількості документів.
- ЛСА можна застосовувати як з навчанням так і без нього.

1.5 Недоліки ЛСА

- Цей метод працює значно повільніше при збільшенні кількості вхідних даних. Десь N^{2k} , де $N = N_{\text{doc}} + N_{\text{term}}$ – кількість документів та термів, k – розмір простору факторів.
- Стохастична модель не співпадає з реальністю.

РОЗДІЛ 2. ОПИС РОБОТИ ЛСА НА ПРИКЛАДІ КЛАСИФІКАЦІЇ СТАТЕЙ

2.1 Підготування даних

Для прикладу було обрано декілька статей про SpaceX, нафту та карантин (Додаток А). Їх обрано не випадково, тому що інакше потрібен великий об'єм даних, що значно ускладнило б подальші обчислення.

Далі, потрібно підготувати статті.

- 1) Слід видалити усі слова, без змістового навантаження, наприклад прийменники, частки. Це слова, які зустрічаються в майже усіх текстах. Так роблять і пошукові агрегатори, наприклад google.
- 2) Далі потрібно зробити морфологічний пошук, процес пошуку основи слова. Це робиться не обов'язково, а залежить від мови, якою написані статті. Тобто якщо у мові немає багато форм одного слова. Наприклад, англійська. Також, це можна не робити, якщо обсяг статей дуже великий, щоб не ускладнювати обчислення.
- 3) Можна видалити слова, які зустрічаються тільки один раз у всіх статтях, або слова, які зустрічаються тільки в одній статті. Це потрібно зробити, щоб спростити математичні обчислення, тому що це не сильно вплине на результат.

Після підготовчих дій вийдуть такі тексти (Додаток Б).

2.2 Побудова матриці вживаності

Для ЛСА потрібно будувати частотні матриці (вживаності), яка описує кількість слів, яка зустрілась у тексті. Нехай рядки це слова, які зустрічаються у тексті, стовпчики – тексти, а елементи перетину – це кількість повторювань слова у статті.

Таблиця 1 – Матриця вживаності

	C1	C2	C3	C4	C5	C6	C7	C8
супутник	1	0	0	0	1	0	0	1
тисяч	0	1	1	0	0	0	0	0
інтернет	0	0	0	0	1	0	0	1
україн	0	0	0	2	0	0	1	0
опек	0	1	0	0	0	1	0	0
зараж	0	0	1	0	0	0	1	0
спасех	1	0	0	0	1	0	0	1
нафт	0	1	0	2	0	1	0	0
планет	0	0	0	0	1	0	0	1
коронавірус	0	0	1	0	0	0	1	0

2.3 Розклад матриці

Найбільш поширений варіант латентно-семантичного аналізу це розклад матриці. Найчастіше використовуваний варіант - це SVD (сингулярний розклад). За допомогою цього методу можна розкласти одну матрицю у декілька ортогональних і їх лінійна комбінація буде дуже точно приближена до початкової матриці.

Посилаючись до теореми о сингулярном розкладі матриць[6], матриця може бути представлена як додток трьох матриць:

$$A = U\Sigma V^t, \quad (1)$$

де Σ – це діагональна матриця $m \times n$ з додатних елементів, які є сингулярними числами;

U – це матриця $n \times m$, яка складається з лівих сингулярних векторів;

V – це матриця $n \times n$, яка складається з правих сингулярних векторів;

V^t – це транспонована матриця до V

Таблиця 2 – Матриця U

супутник	0	0.557	0	0	0.435	0	0	-0.677
тисяч	0.144	0	0.303	-0.543	0	-0.713	0.289	0
інтернет	0	0.435	0	0	-0.557	0	0	0.1
україн	0.616	0	0.098	0.554	0	-0.033	0.551	0
опек	0.197	0	-0.128	-0.601	0	0.622	0.443	0
зараж	0.124	0	0.624	-0.022	0	0.22	-0.214	0.176
спасех	0	0.557	0	0	0.435	0	0	0.677
нафт	0.728	0	-0.321	-0.191	0	-0.074	-0.57	0
планет	0	0.435	0	0	-0.557	0	0	-0.1
коронавірус	0.124	0	0.624	-0.022	0	0.22	-0.214	-0.176

Таблиця 3 – Матриця V^t

C1	C2	C3	C4	C5	C6	C7	C8
0	0.336	0.123	0.845	0	0.291	0.272	0
0.369	0	0	0	0.657	0	0	0.657
0	-0.068	0.72	-0.207	0	-0.209	0.625	0
0	-0.71	-0.313	0.386	0	-0.421	0.27	0
0.929	0	0	0	-0.261	0	0	-0.261
0	-0.211	-0.348	-0.273	0	0.699	0.521	0
0	0.578	-0.498	-0.142	0	-0.454	0.438	0
0	0	0	0	-0.707	0	0	0.707

Таблиця 4 – Матриця Σ

3.182	0	0	0	0	0	0	0
0	3.02	0	0	0	0	0	0
0	0	2.154	0	0	0	0	0
0	0	0	1.881	0	0	0	0
0	0	0	0	0.936	0	0	0
0	0	0	0	0	0.784	0	0
0	0	0	0	0	0	0.28	0
0	0	0	0	0	0	0	0

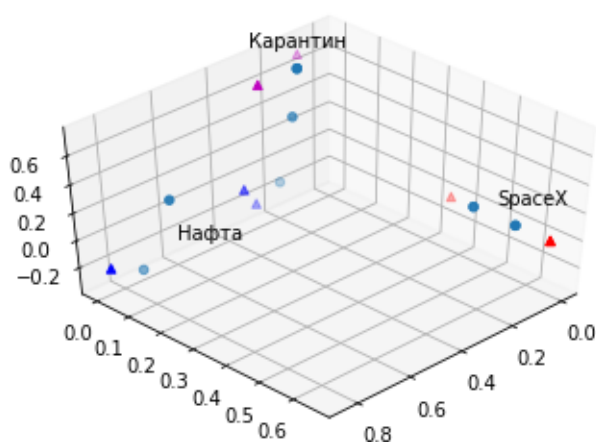
Для розрахунків використовувалась бібліотека TensorFlow на Python.
Код розрахунку(Додаток В) .

2.4 Обробка результатів

Перевагою сингулярного розкладу є те, що цей метод відокремлює ключові складові матриці. Тобто знаючи правила добутку матриці і подивившись на діагональну матрицю можна зрозуміти, які елементи дають значний вклад. Тому ми можемо відкинути деякі стовбці матриці U та строки матриці V^t .

Скільки відкидати залежить від кількості текстів. Для даного розміру залишим тільки перші 3(Додаток Г).

Відмітимо на графіку точки:
червоним кольором – статті про SpaceX;
синім кольором – про нафту;
рожевим кольором – про карантин;
зеленим кольором – усі слова.



На графіку видно, що слова і статті зібрались у три групи. І якщо подивитись на зміст статей та слова, то стає зрозуміло, що ці слова та статті відносяться до однієї теми.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ЛАТЕНТНО-СЕМАНТИЧНОГО АНАЛІЗУ

3.1 Огляд засобів програмної розробки

Для розробки додатку було використано мову програмування Python 3.

Python 3[8] – це інтерпретована мова програмування високого рівня. Мова має строгу динамічну типізацію. Мова підтримує багато парадигм: об'єктно-орієнтовану, функціональну, структурну.

Головними перевагами цієї мови є:

- швидкість, тому що основні бібліотеки написані на C++;
- дуже велика кількість сторонніх бібліотек для роботи з математичним обчисленнями, машиним навчанням тощо;
- більшість програм написаних на Python будуть працювати майже на усіх ОС;
- простий синтаксис.

Додатково були використані бібліотеки.

Numpy[9] – найпопулярніша бібліотека на Python для математичних обчислень, обчислень матриць та багатовимірних списків. Аналогів з подібними можливостями та підтримкою не має.

TensorFlow[7] – відкрита програмна бібліотека, яку розробила компанія Google. Використовується для розробки та тренування нейронних мереж Її використовує і сама компанія Google. Основною мовою програмування є Python, але є можливість використовувати TensorFlow й на Java, C#, Haskell, C++, Swift, Go.

Natural Language Toolkit[10] – набір бібліотек для обробки мови природного характеру. Є багато прикладів даних на багатьох мовах англійський, німецький, французькій, російській тощо. Має дуже добру документацію.

Matplotlib[12] – бібліотека, яка дозволяє візуалізувати дані, як у 2D так й у 3D.

SciPy[15] – бібліотека на Python, яка дозволяє виконувати наукові та інженерні розрахунки.

Був використан Google Colab, як середовище розробки.

Google Colab[11] – це веб-сервіс від компанії Google, який дозволяє запускати Jupyter Notebook. У ньому дуже зручно програмувати на Python. Особливо, якщо ви працюєте з нейронними мережами. Тому що там можна увімкнути функцію, яка дозволить запускати свої тренування моделей на сервері Google використовуючи GPU. Крім того Jupyter Notebook дозволяє запускати окремі частини коду та робити зручні помітки.

3.2 Опис реалізації розробки

Як вже зазначалось, уся розробка була зроблена у Google Colab. Тому роботу розбито на частини. Як це було зроблено у Розділі 2.

1) Імпорт усіх необхідних бібліотек.

```
import tensorflow as tf
import numpy as np
import glob
from collections import Counter
import nltk
import itertools
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.spatial.distance import pdist
from scipy.cluster import hierarchy
```

glob – модуль для роботи з файлами. З них зчитувались статті.

collections – модуль для роботи з додатковими колекціями

itertools – модель, у якому зберігаються корисні ітератори.

2) Константи

```
path = '/content/drive/My Drive/articles/*.txt'
stopwords = ['один', 'два', 'три', 'чотири', 'п'ять',
symbols = ['\n', '\t', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '!', '@', '#', '$', '%', '&', '*', '^', '&#x2D;', '&#x26;', '&#x27;', '&#x28;', '&#x29;', '&#x3A;', '&#x40;', '&#x41;', '&#x42;', '&#x43;', '&#x44;', '&#x45;', '&#x46;', '&#x47;', '&#x48;', '&#x49;', '&#x4A;', '&#x4B;', '&#x4C;', '&#x4D;', '&#x4E;', '&#x4F;', '&#x50;', '&#x51;', '&#x52;', '&#x53;', '&#x54;', '&#x55;', '&#x56;', '&#x57;', '&#x58;', '&#x59;', '&#x5A;', '&#x5B;', '&#x5C;', '&#x5D;', '&#x5E;', '&#x5F;', '&#x60;', '&#x61;', '&#x62;', '&#x63;', '&#x64;', '&#x65;', '&#x66;', '&#x67;', '&#x68;', '&#x69;', '&#x6A;', '&#x6B;', '&#x6C;', '&#x6D;', '&#x6E;', '&#x6F;', '&#x70;', '&#x71;', '&#x72;', '&#x73;', '&#x74;', '&#x75;', '&#x76;', '&#x77;', '&#x78;', '&#x79;', '&#x7A;', '&#x7B;', '&#x7C;', '&#x7D;', '&#x7E;', '&#x7F;', '&#x80;', '&#x81;', '&#x82;', '&#x83;', '&#x84;', '&#x85;', '&#x86;', '&#x87;', '&#x88;', '&#x89;', '&#x8A;', '&#x8B;', '&#x8C;', '&#x8D;', '&#x8E;', '&#x8F;', '&#x90;', '&#x91;', '&#x92;', '&#x93;', '&#x94;', '&#x95;', '&#x96;', '&#x97;', '&#x98;', '&#x99;', '&#x9A;', '&#x9B;', '&#x9C;', '&#x9D;', '&#x9E;', '&#x9F;', '&#xA0;', '&#xA1;', '&#xA2;', '&#xA3;', '&#xA4;', '&#xA5;', '&#xA6;', '&#xA7;', '&#xA8;', '&#xA9;', '&#xAA;', '&#xAB;', '&#xAC;', '&#xAD;', '&#xAE;', '&#xAF;', '&#xB0;', '&#xB1;', '&#xB2;', '&#xB3;', '&#xB4;', '&#xB5;', '&#xB6;', '&#xB7;', '&#xB8;', '&#xB9;', '&#xBA;', '&#xBB;', '&#xBC;', '&#xBD;', '&#xBE;', '&#xBF;', '&#xC0;', '&#xC1;', '&#xC2;', '&#xC3;', '&#xC4;', '&#xC5;', '&#xC6;', '&#xC7;', '&#xC8;', '&#xC9;', '&#xCA;', '&#xCB;', '&#xCC;', '&#xCD;', '&#xCE;', '&#xCF;', '&#xD0;', '&#xD1;', '&#xD2;', '&#xD3;', '&#xD4;', '&#xD5;', '&#xD6;', '&#xD7;', '&#xD8;', '&#xD9;', '&#xDA;', '&#xDB;', '&#xDC;', '&#xDD;', '&#xDE;', '&#xDF;', '&#xE0;', '&#xE1;', '&#xE2;', '&#xE3;', '&#xE4;', '&#xE5;', '&#xE6;', '&#xE7;', '&#xE8;', '&#xE9;', '&#xEA;', '&#xEB;', '&#xEC;', '&#xED;', '&#xEE;', '&#xEF;', '&#xF0;', '&#xF1;', '&#xF2;', '&#xF3;', '&#xF4;', '&#xF5;', '&#xF6;', '&#xF7;', '&#xF8;', '&#xF9;', '&#xFA;', '&#xFB;', '&#xFC;', '&#xFD;', '&#xFE;', '&#xFF;']
```

path – шлях до статей. Вони знаходяться у окремому каталозі. Кожна стаття знаходиться у окремому файлі, який має розширення .txt.

stopwords – слова, які зустрічаються дуже часто, майже у всіх статтях.

symbols – символи, які не несуть змістовного навантаження.

3) Читання файлів та їх запис до списку.

```
articles = []
for name in glob.glob(path):
    try:
        with open(name) as f:
            articles.append(f.read())
    except IOError as exc:
        if exc.errno != errno.EISDIR:
            raise
```

Статті, які були записані до списку.

```
['Прототип супутника, який фахівці SpaceX кілька тижнів будували, вибухнув під час тестової заправки рідким азотом. \n', 'Якщо людина з підозрою на коронавірус самовільно покине місце обсервації, то має сплатити штраф від 17 до 31 тисячі гривень. Якщо це призведе до зараження інших людей – може загрожувати тюремне обмеження волі чи ув'язнення до 3 років.\n', 'Сьогодні вночі над Україною пролетіли десятки супутників SpaceX, що належать Ілону Маску. Супутники будуть підтримувати дешевий інтернет, який покриє всю планету.\n', 'Окупаційна влада Криму оголосила на анексованому півострові карантин – там визнали вже понад 20 випадків зараження коронавірусом. Діти з Криму, так само як і з окупованих районів Донбасу, зможуть дивитися відеоуроки для українських шкіл в етері телеканалу «Дом».\n', 'Дональд Трамп обіцяв накласти тарифи на імпорту нафти, якщо Росія та ОПЕК не домовляться скоротити її виробництво.\n', 'SpaceX успішно вишлю на орбіту Землі ракету Falcon 9 і партію супутників для проекту Starlink, який повинен забезпечити планету глобальним інтернетом. \n', 'На світовому ринку рекордне падіння цін на нафту. Для України це добре, бо нафту ми переважно імпортуємо. Але в цих умовах падають ціни на продукцію, яку Україна продає, через що також слабшає гривня.\n', 'Катар вийде з Організації країн експортерів нафти з січня 2019 – країна була членом ОПЕК 57 років та видобувала 610 тисяч барелів на день. \n']
```

4) Підготування статей.

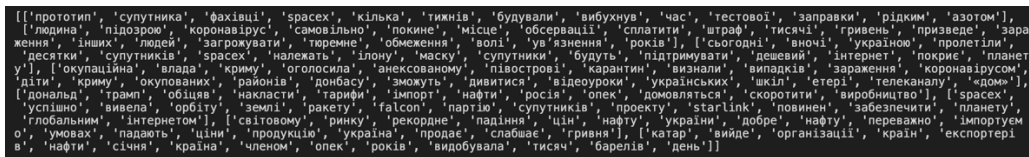
Змінюємо усі великі літери на маленькі та видаляємо зі статей усі символи, які не мають змісту.

```
filteredArticles = []
for article in articles:
    filteredArticles.append("".join(filter(lambda x: x not in symbols, article.lower())))
```

Видаляємо слова, які не містять змісту. Тут використовується власноручнаписана збірка слів. В бібліотеці NLTK є готові збірки для англійської, російської та інших мов. Але української немає.

```
separArt = []
for article in filteredArticles:
    separArt.append([word for word in article if word not in stopwords])
```

Ось що вийшло.



Далі робимо процедуру морфологічного пошуку. Для цього використовуємо алгоритм Портера, який винайшов Мартин Портер у 1990 році. У цьому алгоритмі не використовуються ніякі словники, у ньому просто перебирається слово. І відкидається усе зайве. Реалізації цього алгоритма для української мови немає, тому було перероблено реалізацію з російської[13] (Додаток Г).

```
stemmsWords = []
for article in separArt:
    stemmsWords.append([Porter.stem(word) for word in article])
```

Тут видаляємо слова, які зустрічаються тільки один раз.

```
counts = Counter(list(itertools.chain.from_iterable(stemmsWords)))
duplicates = list(set( [id for id in list(itertools.chain.from_iterable(stemmsWords)) if counts[id] > 1]))
duplicates.sort()
```

5) Складання частотної матриці.

Знаходимо індекси статей у яких зустрічаються слова.

```
articlesNum={}

for i in range(0,len(articles)):
    for w in duplicates:
        if w in stemmsWords[i]:
            if w not in articlesNum:
                articlesNum[w]= [i]
            elif w in articlesNum:
                articlesNum[w]= articlesNum[w]+[i]

newDict = articlesNum.copy()
for el in newDict:
    if len(articlesNum[el]) < 2:
        del articlesNum[el]
    duplicates.remove(el)
```

Ось, що вийшло.

```
{'спасех': [0, 2, 5], 'спутник': [0, 2, 5], 'зараженн': [1, 3], 'коронавірус': [1, 3], 'рок': [1, 7], 'тисяч': [1, 7], 'планет': [2, 5], 'україн': [2, 6], 'інтернет': [2, 5], 'нафт': [4, 6, 7], 'опек': [4, 7]}
```

На основі цього складається матриця.

```
freqMatrix = np.zeros((len(articlesNum), len(articles)))
for i, k in enumerate(duplicates):
    for j in articlesNum[k]:
        freqMatrix[i,j] += 1
```

Вивід:

```
for i, k in enumerate(duplicates):
    print(k + ' ' + str(freqMatrix[i]))
```

```
спасех [1. 0. 1. 0. 0. 1. 0. 0.]
зараженн [0. 1. 0. 1. 0. 0. 0. 0.]
коронавірус [0. 1. 0. 1. 0. 0. 0. 0.]
нафт [0. 0. 0. 0. 1. 0. 1. 1.]
опек [0. 0. 0. 0. 1. 0. 0. 1.]
планет [0. 0. 1. 0. 0. 1. 0. 0.]
рок [0. 1. 0. 0. 0. 0. 0. 1.]
спутник [1. 0. 1. 0. 0. 1. 0. 0.]
тисяч [0. 1. 0. 0. 0. 0. 0. 1.]
україн [0. 0. 1. 0. 0. 0. 1. 0.]
інтернет [0. 0. 1. 0. 0. 1. 0. 0.]
```

6) Сингулярний розклад.

Знаходимо сингулярний розклад матриці за допомогою функції `svd` з бібліотеки `TensorFlow`[14].

```
s, u, v = tf.linalg.svd(freqMatrix)
vt = tf.transpose(v)
```

Робимо перевірку. Матриця повинна бути такою самою, як і частотна матриця.

SVD review

$$M = USV^t$$

```
[14] np.round(np.dot(np.dot(u,np.diag(s)),vt))

array([[ 1., -0.,  1., -0.,  0.,  1.,  0., -0.],
       [-0.,  1., -0.,  1., -0., -0., -0., -0.],
       [-0.,  1., -0.,  1., -0.,  0., -0.,  0.],
       [-0.,  0., -0., -0.,  1.,  0.,  1.,  1.],
       [ 0., -0.,  0., -0.,  1.,  0.,  0.,  1.],
       [ 0., -0.,  1., -0., -0.,  1., -0.,  0.],
       [ 0.,  1., -0.,  0.,  0., -0.,  0.,  1.],
       [ 1., -0.,  1., -0.,  0.,  1.,  0., -0.],
       [ 0.,  1., -0.,  0.,  0., -0.,  0.,  1.],
       [ 0.,  0.,  1.,  0.,  0., -0.,  1.,  0.],
       [ 0., -0.,  1., -0., -0.,  1., -0.,  0.]])
```

7) Візуалізація результатів.

Для візуалізації використаємо бібліотеку `Matplotlib` та зменшемо кількість сингулярних векторів і чисел до трьох для того, щоб відобразити це у трьохвимірному графіці.

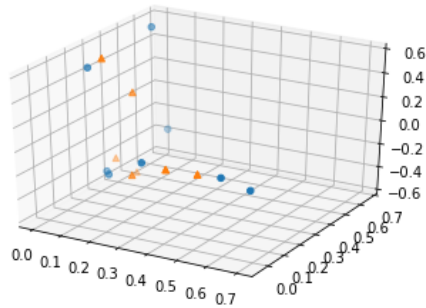
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(vt[0],vt[1],vt[2], marker = 'o')
ax.scatter([el[0] for el in u], [el[1] for el in u], [el[2] for el in u], marker = '^')

ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')

plt.show()
```

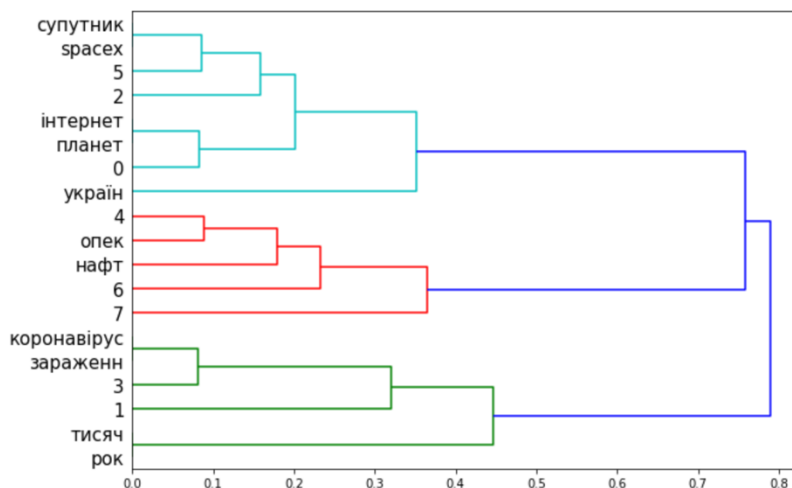
Такий вийшов графік.



Скориставшись бібліотекою SciPy та функцією pdist, ми знайдемо евклідову відстань між точками. Також виведемо дендограму для наявного показу. На ній ми бачимо, що наші статті поділено на 3 категорії.

У Додатку Д можна побачити окремо дендограми для статей та окремо для слів.

```
plt.figure(figsize=(10, 7))
dist = pdist(np.concatenate((u,tf.transpose(vt).numpy()), 'euclidean')
Z = hierarchy.linkage(dist, method='average')
hierarchy.dendrogram(Z, labels=duplicates + [str(x) for x in range(len(articles))], color_threshold=.5,
leaf_font_size=15, count_sort=True, orientation='right')
plt.show()
```



ВИСНОВОК

В ході роботи було детально досліджено латентно-семантичний аналіз. Можна зробити такі висновки: ЛСА є дуже функціональним засобом обробки природної мови, за допомогою нього можна шукати інформацію, знаходити залежності між текстами, фільтрувати їх, класифікувати, оцінювати тощо.

У ході практичної роботи було реалізовано засіб для класифікації статей та досліджена велика кількість програмних технологій, особливо технологій, які стосуються машинному навчанню. Були відібрані тільки кращі продукти, які користуються попитом та вже не один раз показали себе з кращого боку.

СПИСОК ЛІТЕРАТУРИ

1. Automatic Document Classification [Стаття] – Режим доступу до ресурсу: <https://dl.acm.org/doi/10.1145/321160.321165>
2. United States Patent : 4839853 [Електронний ресурс] – Режим доступу до ресурсу:
<http://patft.uspto.gov/netacgi/nphParser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fmetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=4839853.PN.&OS=PN/4839853&RS=PN/4839853>
3. Automating the Assignment of Submitted Manuscripts to Reviewers [Стаття] – Режим доступу до ресурсу:
<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.9793&rep=rep1&type=pdf>
4. Computerized cross-language document retrieval using latent semantic indexing [Електронний ресурс] – Режим доступу до ресурсу:
<https://patents.google.com/patent/US5301109A/en>
5. Latent semantic analysis for text-based research [Електронний ресурс] – Режим доступу до ресурсу:
<https://link.springer.com/content/pdf/10.3758/BF03204765.pdf>
6. Singular Value Decomposition and Least Squares Solutions [Електронний ресурс] – Режим доступу до ресурсу:
https://link.springer.com/chapter/10.1007%2F978-3-662-39778-7_10
7. TensorFlow [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.tensorflow.org/about>
8. Python [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.python.org>
9. Numpy [Електронний ресурс] – Режим доступу до ресурсу:
<https://numpy.org>

- 10.NLTK [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.nltk.org/index.html>
- 11.Google Colab [Електронний ресурс] – Режим доступу до ресурсу:
<https://matplotlib.org>
<https://colab.research.google.com>
- 12.Matplotlib [Електронний ресурс] – Режим доступу до ресурсу:
<https://matplotlib.org>
- 13.Porter algorithm для російської мови [Електронний ресурс] – Режим доступу до ресурсу:
<http://www.algorithmist.ru/2010/12/porter-stemmer-russian.html>
- 14.SVD TensorFlow [Електронний ресурс] – Режим доступу до ресурсу:
https://www.tensorflow.org/versions/r1.15/api_docs/python/tf.linalg/svd
- 15.SciPy [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.scipy.org>
- 16.How much information? [Електронний ресурс] – Режим доступу до ресурсу:
http://web.archive.org/web/20180219162428/https://chnm.gmu.edu/digitalhistory/links/pdf/preserving/8_5a.pdf

Додатки

Додаток А

Статті для аналізу

1. Прототип супутника, який фахівці SpaceX кілька тижнів будували, вибухнув під час тестової заправки рідким азотом.
2. Катар вийде з Організації країн-експортерів нафти з січня 2019 — країна була членом ОПЕК 57 років та видобувала 610 тисяч барелів на день.
3. Якщо людина з підозрою на коронавірус самовільно покине місце обсервації, то має сплатити штраф від 17 до 31 тисячі гривень. Якщо це призведе до зараження інших людей — може загрожувати тюремне обмеження волі чи ув'язнення.
4. На світовому ринку рекордне падіння цін на нафту. Для України це добре, бо нафту ми переважно імпортуємо. Але в цих умовах падають ціни на продукцію, яку Україна продає, через що також слабшає гривня.
5. SpaceX успішно вивела на орбіту Землі ракету Falcon 9 і партію супутників для проекту Starlink, який повинен забезпечити планету глобальним інтернетом.
6. Дональд Трамп обіцяв накласти тарифи на імпорт нафти, якщо Росія та ОПЕК не домовляться скоротити її виробництво.
7. Окупаційна влада Криму оголосила на анексованому півострові карантин — там визнали вже понад 20 випадків зараження коронавірусом. Діти з Криму, так само як і з окупованих районів Донбасу, зможуть дивитися відеоуроки для українських шкіл в етері телеканалу «Дом»
8. Сьогодні вночі над Україною пролетіли десятки супутників SpaceX, що належать Ілону Маску. Супутники будуть підтримувати дешевий інтернет, який покриє всю планету.

Додаток Б

Відформатовані статті

- 1) Прототип **супутника**, який фахівці **SpaceX** кілька тижнів будували, вибухнув під час тестової заправки рідким азотом.
- 2) Катар вийде з Організації країн-експортерів **нафти** з січня 2019 — країна була членом **ОПЕК** 57 років та видобувала 610 **тисяч** барелів на день.
- 3) Якщо людина з підозрою на **коронавірус** самовільно покине місце обсервації, то має сплатити штраф від 17 до 31 **тисячі** гривень. Якщо це призведе до **зараження** інших людей — може загрожувати тюремне обмеження волі чи ув'язнення.
- 4) На світовому ринку рекордне падіння цін на **нафту**. Для **України** це добре, бо **нафту** ми переважно імпортуємо. Але в цих умовах падають ціни на продукцію, яку **Україна** продає, через що також слабшає наша валюта.
- 5) **SpaceX** успішно вивела на орбіту Землі ракету Falcon 9 і партію **супутників** для проекту Starlink, який повинен забезпечити **планету** глобальним **інтернетом**.
- 6) Дональд Трамп обіцяв накласти тарифи на імпорт **нафти**, якщо Росія та **ОПЕК** не домовляться скоротити її виробництво.
- 7) Окупаційна влада Криму оголосила на анексованому півострові карантин — там визнали вже понад 20 випадків **зараження коронавірусом**. Діти з Криму, так само як і з окупованих районів Донбасу, зможуть дивитися відеоуроки для **українських** шкіл в етері телеканалу «Дом»
- 8) Сьогодні вночі над нами пролетіли десятки **супутників SpaceX**, що належать Ілону Маску. **Супутники** будуть підтримувати дешевий **інтернет**, який покриє всю **планету**.

Додаток В

```
import tensorflow as tf
import numpy as np
import pandas as pd

a = np.matrix(""" 1 0 0 0 1 0 0 1;
                  0 1 1 0 0 0 0 0;
                  0 0 0 0 1 0 0 1;
                  0 0 0 2 0 0 1 0;
                  0 1 0 0 0 1 0 0;
                  0 0 1 0 0 0 1 0;
                  1 0 0 0 1 0 0 1;
                  0 1 0 2 0 1 0 0;
                  0 0 0 0 1 0 0 1;
                  0 0 1 0 0 0 1 0""", dtype='float32')
s, u, v = tf.linalg.svd(a, full_matrices=False)
pd.DataFrame(u.numpy()).to_csv("U.csv", float_format='%.3f')
pd.DataFrame(tf.transpose(v).numpy()).to_csv("V.csv", float_format='%.3f')
pd.DataFrame(s.numpy()).to_csv("S.csv", float_format='%.3f')
```

Додаток Г

Таблиця 5 – Матриця U (зменшена)

супутник	0	0.557	0
тисяч	0.144	0	0.303
інтернет	0	0.435	0
україн	0.616	0	0.098
опек	0.197	0	-0.128
зараж	0.124	0	0.624
спасех	0	0.557	0
нафт	0.728	0	-0.321
планет	0	0.435	0
коронавірус	0.124	0	0.624

Таблиця 6 – Матриця V^t (зменшена)

C1	C2	C3	C4	C5	C6	C7	C8
0	0.336	0.123	0.845	0	0.291	0.272	0
0.369	0	0	0	0.657	0	0	0.657
0	-0.068	0.72	-0.207	0	-0.209	0.625	0

Таблиця 7 – Матриця Σ (зменшена)

3.182	0	0
0	3.02	0
0	0	2.154

Додаток Г

```
import re

class Porter:
    PERFECTIVEGROUND = re.compile(
        "((ив|ивши|ившись)|((?<=[ая])(в|вши|вшись)))$")
    REFLEXIVE = re.compile(u"(с[яьи])$")
    ADJECTIVE = re.compile(
        u"(ими|ій|ий|а|е|ова|ове|ів|є|їй|єє|еє|я|ім|ем|им|ім|их|іх|ою|йми|іми|у|ю|ого|ому|ої)$")
    )
    PARTICIPLE = re.compile(u"(ий|ого|ому|им|ім|а|ій|у|ою|ій|і|их|йми|их)$")
    VERB = re.compile(
        u"(сь|ся|ив|ать|ять|у|ю|ав|али|учи|ячи|вши|ши|е|ме|ати|яти|є)$")
    NOUN = re.compile(
        u""""(а|ев|ов|е|ями|ами|еи|и|ей|ой|ий|й|иям|ям|ием|ем|ам|ом|о|у|ах|иях|ях|ь|ию|ью|ю|ия|ья|я|і|ові|ї|ею|єю|ою|є|єві|ем|єм|ів|їв|ю)$""")
    )
    RVRE = re.compile(u"^.*?[аеиоюяііє])(.*)$")
    DERIVATIONAL = re.compile(
        u"^аеиоюяііє[аеиоюяііє]+[аеиоюяііє].*(?<=і)сть?$")
    DER = re.compile(u"ість?$")

    def stem(word):
        word = word.lower()
        m = re.match(Porter.RVRE, word)
        if m and m.group(1):
            pre = m.group(1)
            rv = m.group(2)
            temp = Porter.PERFECTIVEGROUND.sub('', rv, 1)
            if temp == rv:
                rv = Porter.REFLEXIVE.sub('', rv, 1)
                temp = Porter.ADJECTIVE.sub('', rv, 1)
                if temp != rv:
                    rv = temp
                    rv = Porter.PARTICIPLE.sub('', rv, 1)
            else:
                temp = Porter.VERB.sub('', rv, 1)
                if temp == rv:
                    rv = Porter.NOUN.sub('', rv, 1)
                else:
                    rv = temp
            else:
                rv = temp

            if re.match(Porter.DERIVATIONAL, rv):
                rv = Porter.DER.sub('', rv, 1)

            word = pre + rv
        return word

    stem = staticmethod(stem)
```

Додаток Д

