МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти)

на тему Методи оцінювання подібності текстових документів про аналіз руйнувань внаслідок війни

Виконав: студент(ка) 2 курсу, групи КНТ-811м

Спеціальності 124 – Системний аналіз
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Інтелектуальні технології та прийняття рішень в складних системах»

Кондратов Д.О.
(прізвище та ініціали)
Керівник Бакурова А.В.
(прізвище та ініціали)
Рецензент Лозовська Л.І.
(прізвище та ініціали)

2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Кафедра Системного аналізу та обчислювальної математики
Ступінь вищої освіти магістр
Спеціальність 124 – Системний аналіз
(код і найменування)
Освітня програма
(спеціалізація) «Інтелектуальні технології та прийняття рішень в складних системах»
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
_____Г.В.Корніч_____
«_____» _____20___року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Кондратова Дмитра Олександровича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Методи оцінювання подібності текстових документів про аналіз руйнувань внаслідок війни
керівник проєкту (роботи) Бакурова Анна Володимирівна д-р екон. наук, проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від « 28 » листопада 2022 року № 407
2. Строк подання студентом проєкту (роботи) « 19 » грудня 2022 року
3. Вихідні дані до проєкту (роботи) Онтологія, концептуальні графи, коефіцієнт подібності та 6 проаналізованих джерел
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) В першому розділі проаналізовані роботи авторів, що розглядали методи порівняння документів на подібність. В другому розділі розглянуто основні методи оцінювання подібності документів та їх проблематика. проаналізовані оптимальні розв'язки другої задачі з варіюванням об'єму згідно нормальному та рівномірному законам розподілу. В третьому розділі проведено аналіз статей іноземних видань на тему руйнувань під час війни в Україні. В четвертому розділі зроблено основні розрахунки за методами, побудовано онтологію для методу концептуальних графів та розроблено програму для методу латентно-семантичного індексування. В п'ятому розділі розроблено програму для методу за коефіцієнтом Дайса на мові Python. Проведено аналіз результатів
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1	Бакурова А.В., д-р екон. наук, проф.		
2	Бакурова А.В., д-р екон. наук, проф.		
3	Бакурова А.В., д-р екон. наук, проф.		
Нормокон троль	Широкорад Д.В., к.ф.-м.н., ст. викл.		

7. Дата видачі завдання «02» вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Сформулювати мету на основні завдання дипломної роботи	02.09.2022 – 09.09.2022	
2	Опрацювати роботи авторів за темою дослідження	12.09.2022 – 30.09.2022	
3	Розробка застосунку для розв’язку задачі	03.10.2022 – 31.10.2022	
4	Адаптування застосунку	01.11.2022 – 11.11.2022	
5	Розрахунки та аналіз даних	12.11.2022 – 17.11.2022	
6	Оформлення пояснювальної записки	18.11.2022 – 13.12.2022	
7	Попередній захист дипломної роботи та отримання рецензій	13.12.2022 – 18.12.2022	
8	Захист дипломної роботи	19.12.2022	

Студент(ка) _____ Кондратов Д.О.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Бакурова А.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

ПЗ: 59 с; 22 рис; 2 додатки; 18 джерел

Об’єкт дослідження – методи оцінювання подібності електронних документів.

Предмет дослідження – коефіцієнт подібності статті іномовного видання.

Мета роботи – аналіз методів оцінювання подібності статей іноземних видань про руйнування внаслідок війни, побудова онтології методів оцінки руйнувань внаслідок війни та створення програмного забезпечення на базі одного з методів.

Методи дослідження – аналітичний, програмування на мові Python.

В дипломній роботі було проведено аналіз методів оцінювання подібності статей іноземних видань («New York Times», «The Guardian») про руйнування під час війни в Україні та розрахунки за допомогою обраних методів. Побудовано онтологію методів оцінки руйнувань внаслідок війни та створено програмне забезпечення на базі одного з методів.

Ключові слова: PYTHON, COLLAB, ЛАТЕНТНО-СЕМАТИЧНЕ ІНДЕКСУВАННЯ, МЕТОД АДАПТИВНИХ ОНТОЛОГІЙ, МЕТОД ДАЙСА, МЕТОД ЖАККАРА, СТАТТЯ, КОЕФІЦІЄНТ ПОДІБНОСТІ.

ЗМІСТ

Завдання……..……2
РЕФЕРАТ……...4
Вступ ……6
1 Огляд літератури та останніх досліджень і публікацій……...……8
2 Методи оцінювання подібності текстових документів та їх проблематика...11
2.1 Синтаксичний аналіз……..…..11
2.2 Лексичний аналіз……..…13
2.3 Морфологічний аналіз……..…14
2.4 Семантичний аналіз……..17
2.5 Проблеми обробки текстової інформації……....20
3 Обґрунтування вибору статей для аналізу…….22
4 Дослідження ефективності методів оцінювання подібності електронних текстових документів……..28
4.1 Коефіцієнт Дайса……...28
4.2 Коефіцієнт Жаккара……...29
4.3 Латентно-семантичне індексування…….30
4.4 Метод адаптивних онтологій(концептуальних графів)…….36
4.5 Порівняння методів оцінювання подібності статей……...44
5 Розробка програмного забезпечення……..46
5.1 Обґрунтування вибору середовища та мови реалізації……46
5.2 Опис програмного застосунку……..47
Висновки…..……..51
Перелік посилань…….52
Додаток А Код програми за методом латентно-семантичного індексування ...54
Додаток Б Код програми за методом Дайса …….57

ВСТУП

Показники схожості тексту відіграють важливу роль у текстових дослідженнях та додатках у таких завданнях, як пошук інформації, класифікація текстів, кластеризація документів, виявлення теми, відстеження теми, генерація питань, питання-відповідь, оцінка есе, оцінка коротких відповідей, машинний переклад, реферування тексту та інше.

У зв'язку з постійно зростаючими обсягами інформаційних ресурсів доступ користувачів до того, що їх цікавить стає дедалі складнішим. Для вирішення цієї проблеми створюються сучасні інформаційні технології, що базуються на потужному фундаменті телекомунікаційних та обчислювальних засобів. Нині ці засоби досягли високого рівня розвитку, особливо яскраво ці успіхи виявляються в галузі розвитку засобів зв'язку та розробки потужних обчислювальних систем. На тлі цих успіхів успіхи в області смислової обробки інформації менш значні. Ці успіхи залежать, перш за все, від досягнень у вивченні процесів людського мислення, процесів мовного спілкування між людьми та від вміння моделювати ці процеси на ЕОМ. Основною проблемою, що виникає при обробці текстової інформації, є труднощі автоматичного складання формалізованого опису смислового змісту документів і, як наслідок цього - труднощі встановлення смислового зв'язку між різними документами [1].

Актуальність досліджуваної проблеми підтверджується досить великою кількістю публікацій, у яких розглядаються різні підходи, методи реалізації та конкретні інструментальні засоби пошуку подібності в різних документах, текстах, статтях та наукових публікацій.

Наукова новизна роботи полягає в тому, що в ній побудовано модель онтології методів оцінки руйнувань внаслідок війни в Україні в середовищі Protege на основі аналізу публікацій іноземних видань, публікацій з аналізом руйнувань внаслідок війни в Україні.

У роботі розглядаються в теоретичному та практичному плані такі питання:

– огляд останніх наукових публікацій на тему дослідження;

– опрацювання основних та найновіших методів аналізу природньої мови та методів дослідження подібності документів: латентно-семантичне індексування, метод зважених концептуальних графів, коефіцієнт Жаккара, коефіцієнт Дайса;

– аналіз публікацій іноземних видань з аналізом руйнувань внаслідок війни в Україні;

– побудова онтології для методу концептуальних графів;

– порівняння методів подібності;

– розробка програмного забезпечення для методів латентно-семантичного індексування та методу за коефіцієнтом Дайса.

1 ОГЛЯД ЛІТЕРАТУРИ ТА ОСТАННІХ ДОСЛІДЖЕНЬ І ПУБЛІКАЦІЙ

Намагаючись виміряти вплив подібних заходів на кластеризацію в мережі (категоризацію веб-документів), Стрехль, Гот, Лейнс оцінили ефективність чотирьох заходів подібності (косинусна подібність, евклідова відстань, індекс Жаккара та кореляція Пірсона) на кількох алгоритмах кластеризації. Задумка, що стоїла за цим дослідженням, полягала в тому, що точний вимір подібності призводить до кращої кластеризації. Експериментальні результати дослідження Стрехль та ін. показали, що косинусна подібність і подібність Жаккара працюють найкраще, а евклідова відстань — найменше [1].

Уайт і Хосе оцінили ефективність восьми показників подібності залежно від того, наскільки добре вони можуть класифікувати документи на теми. Результати експерименту Уайта і Хосе показують, що коефіцієнт кореляції перевершив інші показники, видавши прогнози (подібність до тем), які більш точно відповідали людському судженню. Намагаючись отримати семантичну схожість з текстових документів, Любесик та ін. експериментували з вісьмома різними заходами подібності і виявили, що розбіжність Дженссена-Шеннона, манхеттенська відстань (L1) і евклідова відстань (L2) працюють найкраще, перевершуючи стандартні IR-заходи, такі як косинус і подібність Жаккара [2].

Хуанг розширив роботи Стрехль шляхом включення додаткової міри подібності (розбіжність Кульбака – Лейблера) та використання алгоритму кластеризації k-середніх з n = 1. Результати експериментів Хуанга показують, що кластеризація на основі кореляції Пірсона та розбіжності Кульбака-Лейблера була більш точною, тоді як кластеризація документів на Основою кореляції Пірсона і подібності Жаккара була більш зв'язкова (де зв'язність означає схожість або близькість між членами кластера). Найгірші результати було отримано на евклідовій відстані [3]. У пізнішому дослідженні Форсайт і Шарофф запропонували як зразок підхід до вимірювання ефективності подібних заходів

у порівнянні з людським судженням. Результати, отримані в результаті їх дослідження, показали, що кореляція Пірсона перевершує стандартні заходи, такі як косинус та дивергенція Кульбака-Лейблера (КЛД). Це дослідження відрізняється від попереднього дослідження тим, що воно зосереджено на оцінці показників подібності різних рівнях між текстової подібності. Це з тим, що міра подібності, ефективна на одному рівні текстової подібності, то, можливо неефективна іншому. Крім того, у наведених вище дослідженнях не враховувався вплив методів зважування термінів та моделей подання документів на продуктивність. Це фактори оптимізації, які слід ретельно обирати. Знання того, який метод зважування термінів та моделі подання документа використовувати з конкретним заходом подібності на конкретному рівні між текстової подібності, є важливим для продуктивності пошуку [4].

З наукових публікацій та книг України за даним напрямком можна виділити «Методи та засоби опрацювання інформаційних ресурсів на основі онтологій» Литвина та ін..У книзі розглядається важлива науково-технічна проблема моделювання та проектування інтелектуальних систем управління інформаційними ресурсами на основі онтологій. Для вирішення цієї проблеми розглядається багато методів рішення, зокрема розглядається класифікація інтелектуальних систем для роботи з інформаційними продуктами, розробляється формальна модель такої системи, її окремих складових, інформаційних ресурсів, методи та алгоритми проектування інтелектуальних систем опрацювання інформаційних ресурсів [5].

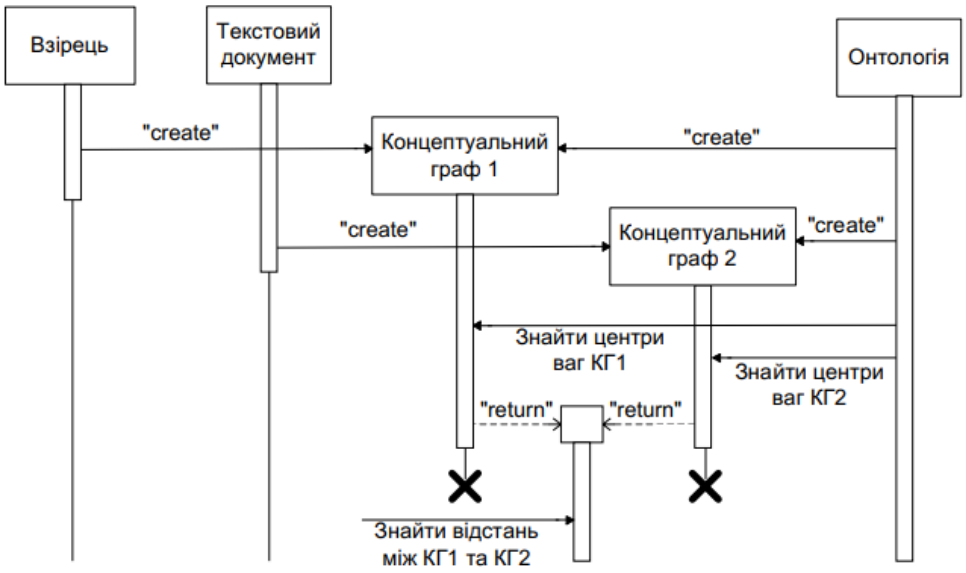

Рисунок 1.1 – Схема семантичного порівняння документів [5]

Також з робіт можна виділити публікацію Даревича у співавторстві з Литвином та Досиним «Оцінка подібності текстових документів на основі визначення інформаційної ваги елементів бази знань», в якій більш детально вивчається та розглядається метод зважених концептуальних графів при роботі з базою знань та більш докладно описується метод описаний в книзі [6]. Окремо Литвином у публікації «Метод оцінювання подібності документів, доповнених контекстом з онтологій» було розглянуто метод на основі введення метрики між концептуальними графами, що є відповідними моделями цих документів [7].

2 МЕТОДИ ОЦІНЮВАННЯ ПОДІБНОСТІ ТЕКСТОВИХ ДОКУМЕНТІВ ТА ЇХ ПРОБЛЕМАТИКА

2.1 Синтаксичний аналіз

Синтаксичний аналіз займає одне з найважливіших місць у ланцюжку обробки текстів природною мовою. Серед центральних завдань комп'ютерної лінгвістики синтаксичний аналіз є «вузьким місцем», оскільки досі не запропоновано достатньо ефективного як з точки зору складності обчислень, так і з точки зору якості обробки підходу для його проведення. Тому на сьогоднішній день синтаксичний аналіз є актуальною та активно досліджуваною проблемою комп'ютерної лінгвістики.
Завданням синтаксичного аналізу є експліцитний опис синтаксичної структури тексту. Більшість моделей уявлення синтаксичної структури спираються або на граматику залежностей, або на граматику безпосередньо-складових. Граматика залежностей передбачає, що речення тексту є деревами залежностей, у яких слова пов'язані орієнтованими дугами, що позначають синтаксичне підпорядкування . Вважається, що цей формалізм добре відображає специфіку мов із довільним порядком слів, у яких між словами може бути присутня значна кількість непроективних зв’язків. До таких мов відноситься німецька, чеська, російська, а також інші східнослов'янські мови.

У граматиці безпосередньо складових речення тексту подаються у вигляді ієрархії складових (синтаксичних груп): вся пропозиція розбивається на проекти, що не перетинаються, і які у свою чергу складаються з дрібніших груп і т.д. аж до атомарних груп – слів речення. У граматиці складових по суті не допустимі непроективні синтаксичні відносини між словами. Тому цей формалізм вважається придатним для мов з фіксованим порядком слів, наприклад, для англійської, де проективність дотримується суворо. В англійській мові зустрічаються і непроективні зв'язки: «John saw a dog yesterday which was a Yorkshire Terrier». У цьому реченні перетинаються зв'язки між «saw – yesterday»

та «dog – was». Тим не менш, незважаючи на низку винятків для більшості конструкцій в англійській властивість проективності дотримується. Слід зазначити, що ці два формалізму застосовуються для синтаксичного аналізу різних мов приблизно однаково. У випадку, коли всі зв'язки в дереві залежності є проективними, воно є ізоморфним відповідним деревом складових. Таким чином, у якихось завданнях можна, не сильно пожертвувавши повнотою, враховувати лише проективні зв'язки та з однаковою ефективністю застосовувати обидва формалізми [8].

Синтаксичний аналіз поділяють на глибокий (повний) аналіз (deep parsing) та поверхневий (shallow parsing) . Завданням глибокого синтаксичного аналізу є побудова повного синтаксичного дерева речення з максимальною пов'язаністю з урахуванням далеких зв’язків, а також визначення граматичних функцій слів речення (що підлягає, присудку, обставини місця, часу тощо). Для поверхневого синтаксичного аналізу немає чіткого визначення, це поняття поєднує в собі різні підходи, що працюють на рівні синтаксису, спрямовані на побудову неповної (частково пов'язаної) синтаксичної структури тексту різної складності. Поверхневий синтаксичний аналіз охоплює такі завдання як поділ пропозиції на рекурсивно неукладені синтаксичні групи (chunking), сегментацію (виділення в пропозиції різних оборотів та простих пропозицій у складі складного), а також побудова поверхневого синтаксичного дерево. Поверхневі аналізатори можуть будувати пов'язані дерева, які близькі до результатів роботи глибоких синтаксичних аналізаторів. Однак поверхневі аналізатори зазвичай не призначені для встановлення всіх синтаксичних зв'язків у реченні, не враховують дальні зв'язки та не призначені для визначення граматичних функцій слів речення [8, 9]. Подібне спрощення завдання синтаксичного аналізу порівняно з глибоким аналізом дозволяє використовувати обчислювально та алгоритмічно простіші методи. Крім того, в рамках спрощеного завдання (наприклад, для виділення синтаксичних груп) вдається досягти високих показників якості.

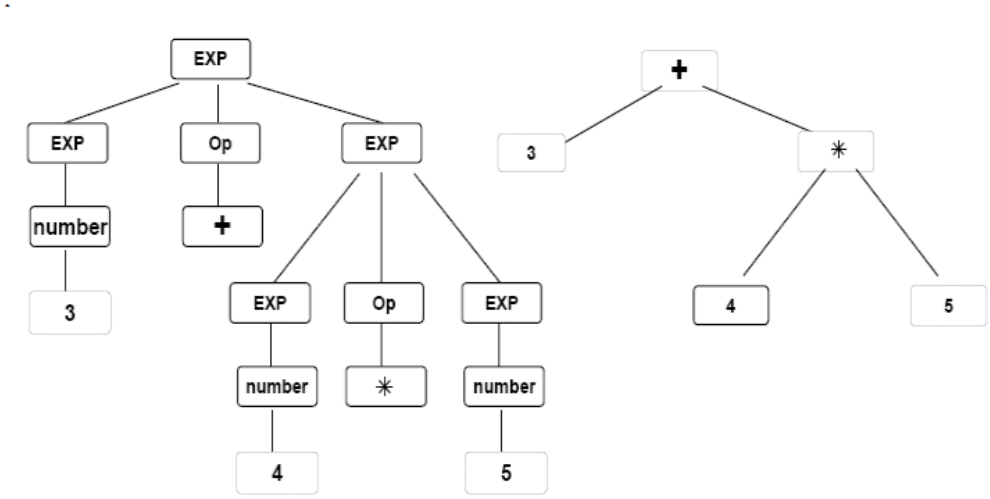

Рисунок 2.1 – Приклад синтаксичного дерева [8]

Методи як повного, і поверхового синтаксичного аналізу знаходять застосування у широкому спектрі різних прикладних завдань обробки текстів. Глибокий синтаксиний аналіз традиційно використовується в системах машинного перекладу, поверхневий аналіз як альтернатива глибокому застосовується в інформаційно-пошукових та інформаційно-аналітичних системах. Крім того, в деяких лінгвістичних аналізаторах поверхневий аналіз є етапом перед обробки тексту перед проведенням глибокого аналізу, що дозволяє в цілому спростити та прискорити процедуру проведення синтаксичного аналізу.

2.2 Лексичний аналіз

Лексичний аналіз — це виділення окремих слів або лексем із вхідного потоку символів і передача відповідних токенів назад до аналізатора.

Оператор повинен мати можливість розпізнавати малі синтаксичні одиниці (токени) і передавати цю інформацію аналізатору. Також потрібно зберігати різні атрибути в таблицях символів або літералів для подальшого використання, наприклад, якщо є змінна, токенізатор згенерує маркер var, а

потім пов’яже ім’я змінної з нею в таблиці символів – у цьому case ім'я змінної є лексемою.

Інші ролі лексичного аналізатора включають видалення пробілів і коментарів і обробку директив компілятора (тобто як препроцесор).

Процес токенізації приймає вхідні дані, а потім пропускає їх через розпізнавач ключових слів, розпізнавач ідентифікаторів, розпізнавач числових констант і розпізнавач констант рядків, кожен з яких подається на власний вихід на основі правил усунення неоднозначності.

Ці правила можуть включати «зарезервовані слова», які не можна використовувати як ідентифікатори (загальні приклади включають початок, кінець, пустоту тощо), таким чином, якщо рядок може бути ключовим словом або ідентифікатором, він вважається ключовим словом. Іншим поширеним правилом є те, що рядок можна інтерпретувати як один лексему або послідовність лексем, зазвичай передбачається перша інтерпретація [10].

Процес лексичного аналізу починається з визначення того, що означає бути лексемою в мові з регулярними виразами чи граматиками, потім це перекладається на абстрактну обчислювальну модель для розпізнавання токенів (недетермінований кінцевий автомат), який потім переведено на реалізовану модель для розпізнавання визначених токенів (детермінований кінцевий автомат), до якої можна зробити оптимізацію (мінімізований DFA).

2.3 Морфологічний аналіз

Морфологічний аналіз (MA) або загальний морфологічний аналіз (GMA) – це метод творчого вирішення багатовимірних, складних проблем шляхом їх системного структурування та вивчення простору всіх можливих рішень, а також ефективний інструмент створення креативних ідей та розробки нових продуктів, технологій та послуг.

Морфологічний аналіз відноситься до добре структурованих евристичних методів, є розширенням методу Списку атрибутів та має спільні риси з технікою Примусових відносин.

Метод ґрунтується на аналізі атрибутів та параметрів системи, генеруванні альтернативних варіантів їх подання, а також створенні та виборі їх нових комбінацій.

Сутність морфологічного аналізу полягає у знаходженні всіх можливих варіантів реалізації об'єкта або вирішення поставленої проблеми шляхом побудови багатовимірних матриць та комбінування їх елементів.

Даний метод заснований на побудові матриці, в якій по вертикальній осі перераховуються всі основні параметри об'єкта, а по горизонтальній - вказується якомога більше варіантів їх реалізації. Комбінування одержаних варіантів елементів об'єкта веде до генерування творчих ідей та рішень [8].

Мета методу - розширити область пошуку нових ідей, шляхом систематичного створення та дослідження якомога більшої кількості комбінацій ознак об'єкта та альтернативних рішень проблеми.

Основні функціями морфологічного методу є:

– Морфологічний аналіз використовується для аналізу та вирішення складних, практичних проблем та знаходження найкращих рішень серед найрізноманітніших можливостей.

– Даний метод найбільш ефективний при вирішенні прикладних проблем та конструкторських розробок, проектуванні нових продуктів, товарів та послуг.

Загальний морфологічний аналіз (GMA) – методологічний підхід, узагальнене ставлення до реальності, особливий спосіб бачення та стиль мислення. Морфологічний аналіз створює основу для системного мислення, спрямованого на виявлення ключових структурних параметрів і всієї різноманітності варіантів їх представленості, а також вибору найбільш корисних комбінацій або візерунків ознак. Морфологічний аналіз ґрунтується на концепції універсального зв'язку між усіма об'єктами, подіями та сферами реальності.

Морфологічний аналіз заснований на комбінаторній природі творчості, яка була розкрита та сформульована у ряді концепцій.

З переваг можна виділити:

– Морфологічний аналіз - це добре структурований метод, що дозволяє істотно розширити область пошуку рішення та ввести в поле зору варіанти, які при звичайному розгляді проблеми можуть бути втрачені.

– Застосування морфологічного аналізу дозволяє глибше зрозуміти сутність та структуру досліджуваної проблеми чи об'єкта.

– Метод має системний і цілеспрямований характер, що дозволяє впорядковувати наявну інформацію та генерувати нові творчі ідеї вирішення проблем.

– Морфологічний аналіз дозволяє розвивати вже існуючі ідеї та вдосконалювати об'єкти, а також генерувати новизну та створювати нові технології, об'єкти та послуги.

– Принцип морфологічного аналізу легко реалізується за допомогою комп'ютерних засобів та може бути використаний у обчислювальній творчості (computational creativity), моделях штучного інтелекту, струкутрно-параметричного синтезу, параметричної архітектури та дизайну.

З недоліків та обмежень методу можна виділити:

– Механічний підхід до аналізу об'єкта веде до генерування великої кількості варіантів, багато з яких виявляються позбавлені практичного змісту.

– При виборі ключових параметрів об'єкта можуть бути втрачені значні чинники, що може значно знижувати ефективність отриманих рішень.

– Простір розв'язування задач визначається межами побудованої матриці, а процес створення нового відбувається всередині морфологічного ящика.

– Сучасні проблеми реального світу вкрай динамічні, невизначені та неоднозначні, а їх сценарії розвитку часто непередбачувані та лежать поза заданими межами морфологічного простору [9].

– Успіх методу безпосередньо залежить від суб'єктивних процесів оцінки та вибору, а відсутність критеріїв відбору корисних комбінацій та варіантів рішень уможливлює втрати корисних та значущих альтернатив.

– Морфологічний аналіз збільшує можливість отримання цікавого рішення, але не гарантує його.

– Комбінування як інструмент генерації новизни є лише одним із цілого спектру механізмів творчої діяльності. У зв'язку з цим, у цьому методі конвергентного мислення переважає над дивергентним, а раціональний аналіз над інтуїтивним вирішенням проблем та перетворюючою уявою.

2.4 Семантичний аналіз

Семантичний аналіз стосується процесу розуміння природної мови (тексту) шляхом вилучення суттєвої інформації, такої як контекст, емоції та настрої, із неструктурованих даних. Це дає комп’ютерам і системам можливість розуміти, інтерпретувати та отримувати значення з речень, абзаців, звітів, реєстрів, файлів або будь-якого документа подібного роду.

Семантичний аналіз аналізує граматичний формат речень, включаючи розташування слів, фраз і речень, щоб визначити зв’язок між незалежними термінами в конкретному контексті. Це найважливіше завдання систем обробки природної мови (NLP). Це також ключовий компонент кількох доступних сьогодні інструментів машинного навчання, таких як пошукові системи, чат-боти та програмне забезпечення для аналізу тексту [11].

Процес семантичного аналізу починається з вивчення та аналізу словникових визначень і значень окремих слів, які також називають лексичною семантикою. Після цього перевіряється зв’язок між словами в реченні, щоб забезпечити чітке розуміння контексту.

Завдяки обробці природної мови та машинному навчанню системи семантичного аналізу мають тенденцію досягати точності на рівні людини. Кілька компаній значною мірою покладаються на інструменти семантичного аналізу, які автоматично отримують цінні дані з неструктурованих даних, таких як електронні листи, звіти клієнтів і відгуки клієнтів.

Метод семантичного аналізу починається з незалежного від мови етапу аналізу набору слів у тексті для розуміння їх значення. Цей крок називається «лексична семантика» і стосується отримання визначення словника для слів у тексті. Згодом слова або елементи аналізуються. Кожному елементу призначається граматична роль, і вся структура обробляється, щоб уникнути будь-якої плутанини, спричиненої неоднозначними словами, які мають кілька значень.

Після семантичний аналіз переходить до етапу інтерпретації, який є критичним для алгоритмів штучного інтелекту. Наприклад, слово «Blackberry» може означати фрукт, компанію чи її продукцію разом із кількома іншими значеннями. Крім того, контекст не менш важливий під час обробки мови, оскільки він враховує середовище речення, а потім надає йому правильне значення.

Таким чином, контекст має життєво важливе значення для семантичного аналізу та вимагає додаткової інформації, щоб призначити правильне значення всьому реченню чи мові.

Технічно семантичний аналіз включає:

– Обробка даних.
– Визначення ознак, параметрів і характеристик оброблених даних
– Представлення даних
– Визначення граматики для аналізу даних
– Оцінка семантичних шарів оброблених даних
– Виконання семантичного аналізу на основі лінгвістичного формалізму.

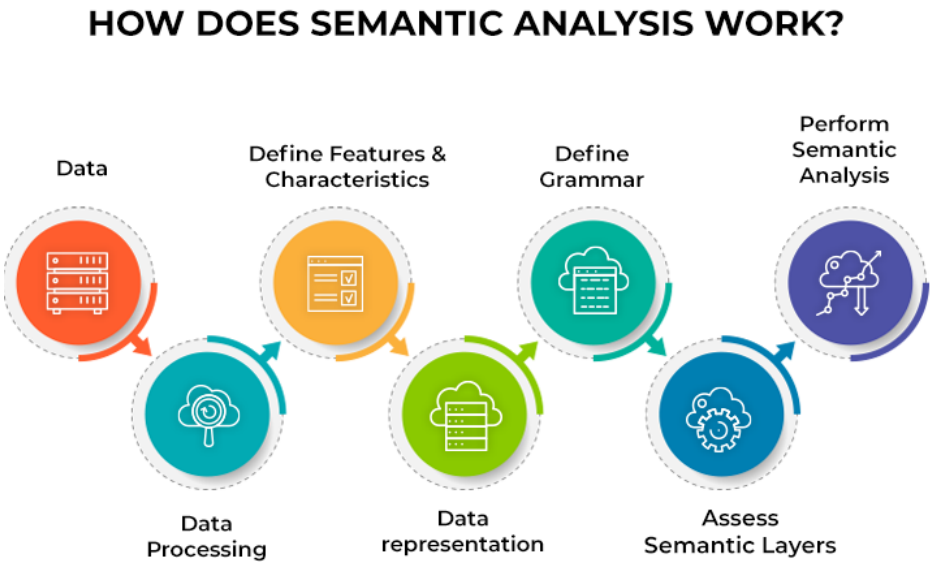

Рисунок 2.2 – Інфографіка як працює семантичний аналіз

Критичні елементи семантичного аналізу є фундаментальними для обробки природної мови:

– Гіпоніми: це відноситься до конкретної лексичної одиниці, яка має зв'язок із більш загальною словесною сутністю, яка називається гіпернімом. Наприклад, червоний, синій і зелений — це гіпоніми кольору, їх гіпернім.

– Мерономіа: відноситься до розташування слів і тексту, які позначають другорядний компонент чогось. Наприклад, манго — це мерономіа мангового дерева.

– Полісемія: це стосується слова, яке має більше одного значення. Однак він представлений одним записом. Наприклад, термін «блюдо» є іменником. У реченні «розкласти посуд на полиці» слово посуд відноситься до різновиду тарілки.

– Синоніми: це стосується слів зі схожим значенням. Наприклад, анотація (іменник) має синоніми анотація–конспект.

– Антоніми: це стосується слів із протилежним значенням. Наприклад, холод має антоніми теплий і гарячий.

– Омоніми: це стосується слів, які мають однакове написання та вимову, але мають зовсім інше значення [8].

2.5 Проблеми обробки текстової інформації

В даний час у зв'язку з постійно зростаючими обсягами інформаційних ресурсів доступ користувачів до того, що їх цікавить стає дедалі складнішим. Для вирішення цієї проблеми створюються сучасні інформаційні технології, що базуються на потужному фундаменті телекомунікаційних та обчислювальних засобів. Нині ці засоби досягли високого рівня розвитку, особливо яскраво ці успіхи виявляються в галузі розвитку засобів зв'язку та розробки потужних обчислювальних систем. На тлі цих успіхів успіхи в області смислової обробки інформації менш значні. Ці успіхи залежать, перш за все, від досягнень у вивченні процесів людського мислення, процесів мовного спілкування між людьми та від вміння моделювати ці процеси на ЕОМ. Основною проблемою, що виникає при обробці текстової інформації, є труднощі автоматичного складання формалізованого опису смислового змісту документів і, як наслідок цього - труднощі встановлення смислового зв'язку між різними документами. Це обумовлено тим, що в різних текстах одні й ті самі ситуації можуть описуватися в термінах різного ступеня спільності та за допомогою різних мовних засобів. І лише людина, яка аналізує документи, керуючись своїми уявленнями про зміст документів та засоби висловлювання цього змісту та спираючись на свої професійні знання та досвід, у стані встановити ступінь смислової близькості аналізованих документів. Більшість систем автоматичної обробки текстової інформації, що функціонують в даний час, не можуть у повному ступені вирішувати ці проблеми. У зв'язку з цим виникає необхідність у розробленні ефективних методів автоматичного аналізу змісту документів. Відмінна особливість запропонованих методів полягає в тому, що вони базуються на

сучасних уявленнях про смислову структуру текстів та оригінальні процедури семантико-синтаксичного та концептуального аналізу.

Тому підсумовуючи можна виділити 7 основних проблем розуміння тексту в обробці природних мов:

– Знання системою контексту та проблемної галузі та навчання цієї системи.

– Різна форма передачі синтаксису (тобто структури) речення у різних мовах.

– Проблема рівнозначності.

– Наявність у тексті нових для комп'ютера слів, наприклад, неологізмів. Самонавчена система повинна вміти «інтуїтивно» визначити (можливо, і неправильно, але з нагодою надалі виправити себе) лексичну роль, морфологічну форму цього слова, спробувати вписати його в існуючу структуру знань, наділити його якимись атрибутами або з'ясувати все це в діалозі з оператором. Система, не здатна до самонавчання, просто втратить якусь кількість інформації.

– Проблема сумісності нової інформації з уже накопиченими знаннями. Нова інформація може якось суперечити вже накопиченої інформації. Необхідно реалізувати механізм, що визначає, у яких випадках потрібно відкинути стару інформацію, а яких – нову.

– Проблема тимчасових протиріч.

– Проблема еліпсів, тобто пропозицій із пропущеними фактично, але існуючими неявно завдяки контексту словами [9].

3 ОБҐРУНТУВАННЯ ВИБОРУ СТАТЕЙ ДЛЯ АНАЛІЗУ

Для проведення дослідження методів оцінювання подібності треба було обрати статті для подальшої роботи. Задля розширення діапазону пошуку основною мовою пошуку було обрано англійську, так як більша частина світу розмовляє та розуміє дану мову, що приводить до того, що кількість видань та статей набагато більша за обраною тематикою чим на українській мові. Також як перевагу можна виділити здебільшого неупередженість до аналітики та прогнозів даних по руйнуванням під час війни. Також для чистоти експерименту було вирішено обирати статті у виданнях різних країн. Найвідомішими у світі виданнями які проводять аналітику глобальних подій можна виділити такі як:

– «New York Times» - американська щоденна газета, що видається у Нью-Йорку з 18 вересня 1851 року. Третя за тиражем газета в країні після USA Today та The Wall Street Journal та 40-та у світі. Веб-сайт «Нью-Йорк таймс» вважається одним із найпопулярніших новинних сайтів із відвідуваністю 30 мільйонів людей на місяць. Слоганом газети є фраза "Всі новини, гідні друку" (All the News That's Fit to Print). Згодом, з появою сайту "Нью-Йорк таймс", цей слоган довелося переробити в "У нас новини, на які ви кликаєте" (All the News That's Fit to Click). Їх стаття «100 Days of War: Death, Destruction and Loss» розповідає про перші дні війни та саму гостру фазу оборони України [12]. Окрім опису загальних подій в статті також проводиться аналітика отриманої шкоди та руйнувань в областях де проводились запеклі бойові дії.

Рисунок 3.1 – Сторінка статті New York Times

– «Forbes» — американський фінансово-економічний журнал, одне з найавторитетніших і найвідоміших економічних друкованих видань у світі. Заснований 15 вересня 1917 року Берті Чарлзом Форбсом. Слоган журналу: "Інструмент капіталіста" - "The Capitalist Tool" [13]. З плюсів і водночас з мінусів даного видання можна виділити більш стислий та тезісний формат подання інформації, та крім приведення статистики та цифр подальша аналітика експертів відсутня.

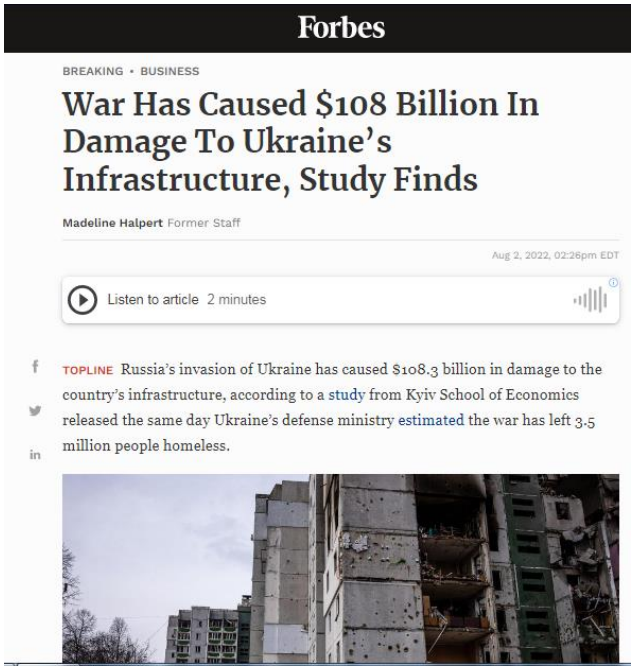

Рисунок 3.2 – Сторінка статті Forbes

– «The Guardian» - щоденна ліволіберальна газета у Великій Британії, заснована в Манчестері в 1821 році під назвою The Manchester Guardian. Сайт газети - найбільш відвідуваний із сайтів британських газет (згідно з даними Alexa), відомих як «якісна преса» (quality press); причому його матеріали можуть відрізнятись від матеріалів паперового видання. Згідно зі статтею в Press Gazette, опублікованій у 2013 році, з усіх британських газет The Guardian було найчитанішим з якісних видань, з урахуванням як продажів паперової версії, так і переглядів сайту газети. У наукових працях газету характеризують як впливову, престижну та якісну, а також як одну з найшанованіших газет у світі. Їх стаття базується на інформації зібраної та обробленої Київською школою економіки з додаванням аналітики та прогнозів на майбутнє автора статті [14].

Рисунок 3.3 – Сторінка статті The Guardian

– «Reuters» - одне з найбільших у світі міжнародних агенцій новин та фінансової інформації, існує з середини XIX століття. У 2008 році куплено корпорацією Thomson, яка після цього стала називатися Thomson Reuters. Діяльність розпочали в Британії, де і знаходиться головний офіс компанії. Стаття «Russia's invasion causes damage across Ukraine» описує та демонструє за допомогою інфографіки завданні збитки України під час війни [15]. Інфографіка є дуже детально пропрацьованою та передає багато інформації. Але попри це для поставленої задачі по аналізу не підходить так як основну інформацію передає за допомогою інфографіки.

Рисунок 3.4 – Приклад інфографіки з статті Reuters

– «United nations news» - Організація Об'єднаних Націй (ООН) — міжнародна організація, створена для підтримки та зміцнення міжнародного

миру та безпеки, а також розвитку співробітництва між державами. ООН вважається універсальним форумом, наділеним унікальною легітимністю, конструкцією міжнародної системи колективної безпеки, що несе, головним елементом сучасної багатосторонньої дипломатії. Штаб-квартира ООН знаходиться у Нью-Йорку; ООН також має додаткові офіси у Відні, Женеві та Найробі. Міжнародний суд ООН перебуває у Гаазі. Основний сайт організації розміщує глобальні новини з можливістю прослухати та прочитати новину на різних мовах учасників ООН. В статті «Ukraine: Cycle of death, destruction, dislocation, and disruption must stop», автор фокусується більш на описі подій чим на аналітиці або прогнозах завданих руйнувань, тому дана стаття підходить лише частково [16].

Рисунок 3.5 – Сторінка статті UN News

– «Rand» - американська некомерційна організація, яка виконує функції стратегічного дослідницького центру, що працює на замовлення уряду США, їх збройних сил і пов'язаних з ними організацій. У статті «Russia's War in Ukraine: Insights from RAND» було здебільшого зібрані прогнози військових аналітиків щодо руйнувань та коротко представлені дані щодо руйнувань на даний час [17].

Рисунок 3.6 – Сторінка статті Rand

Після ознайомлення з статтями було прийнято рішення обрати статті видавництв New York Times – «100 Days of War: Death, Destruction and Loss» та The Guardians – «Russia’s war in Ukraine causing £3.6bn of building damage a week». Саме ці статті більш за все підходять під основні критерії відбору – це англомовні видання, різних політичних поглядів, розташовані в різних країнах та було проведено детальний аналіз руйнувань через війну в Україні.

4 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДІВ ОЦІНЮВАННЯ ПОДІБНОСТІ ЕЛЕКТРОННИХ ТЕКСТОВИХ ДОКУМЕНТІВ

4.1 Коефіцієнт Дайса

Нехай задано документи T_1 і T_2 , тоді коефіцієнт подібності Дайса обчислюють за формулою:

$$d_{(T_1,T_2)} = \frac{2n(T_1 \cap T_2)}{n(T_1) + n(T_2)}$$

де $n(\mathrm{T_1})$ – кількість термінів у документі $\mathrm{T_1}$;

$n(\mathrm{T_2})$ – кількість термінів у документі $\mathrm{T_2}$;

$n(\mathrm{T_1} \cap \mathrm{T_2})$ –кількість спільних термінів у документах $\mathrm{T_1}$ і $\mathrm{T_2}$ відповідно [5].

З отриманих даних проведемо розрахунки:

$$d_{(T_1,T_2)} = \frac{2 * 126}{677 + 317} = \frac{252}{994} = 0{,}25$$

Перевагами використання метода Дайса є швидкість обчислень, простота, нормалізація та обширна область використання. З недоліків можна виділити те, що оцінка однорідності контенту залежить від загальної кількості документів, відсутність впливу семантичного зв’язку та навантаження термінів.

4.2 Коефіцієнт Жаккара

Індекс Жаккара, також відомий як перетин по об'єднанню та Коефіцієнт подібності Жаккара (спочатку присвоєний французькій назві «коефіцієнт комунауті» від Пола Жаккара) – це статистика, яка використовується для оцінки подібності та різноманітність з вибірок наборів. Коефіцієнт Жаккара вимірює подібність між кінцевими вибірковими наборами і визначається як розмір перетину.

Коефіцієнт виражається формулою, що позначає відношення числа термінів, знайдених в двох досліджуваних статтях, до суми термінів, знайдених в статті A, але не знайдених у статті B, і знайдених в статті B, але відсутніх в статті A. Тобто несупадні пари, підсумовані в знаменники, мають вагу більшу вдвічі, порівняно з парами, що збігаються [1]. Коефіцієнт розраховується за даною формулою:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Відстань визначається як:

$$J_\delta(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

Проведемо розрахунки для даного випадку:

$$J(A,B) = \frac{191}{551} = 0{,}35$$

4.3 Латентно-семантичне індексування

Тематичне моделювання автоматично виявляє приховані теми у заданих документах. Це неконтрольований алгоритм аналізу тексту, який використовується для пошуку групи слів у цьому документі. Ця група слів представляє тему. Існує ймовірність того, що один документ може бути пов'язаний із кількома темами. наприклад, група слів, таких як «пацієнт», «лікар», «хвороба», «рак» та «здоров'я» представлятиме тему «охорона здоров'я». Тематичне моделювання - це інший рівень в порівнянні з текстовим пошуком на основі правил, в якому використовуються регулярні вирази.

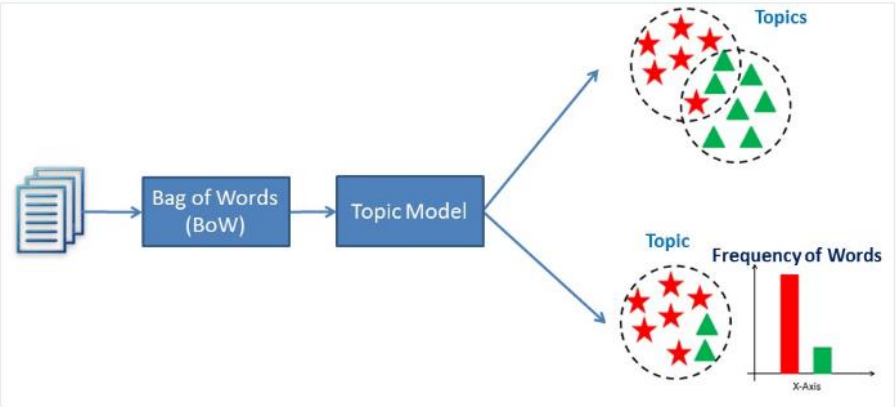

Рисунок 4.1 – Інфографіка процес тематичного моделювання документа

Класифікація тексту — це контрольована проблема машинного навчання, у якій текстовий документ або стаття класифікується за заздалегідь визначеним набором класів. Тематичне моделювання — це процес виявлення груп слів, що зустрічаються разом, у текстових документах. Ці групи пов’язаних слів утворюють «теми». Це форма неконтрольованого навчання, тому набір можливих тем невідомий. Для розв’язання задачі класифікації тексту можна використовувати тематичне моделювання. Моделювання тем визначить теми, представлені в документі, тоді як класифікація тексту класифікує текст в один клас [10].

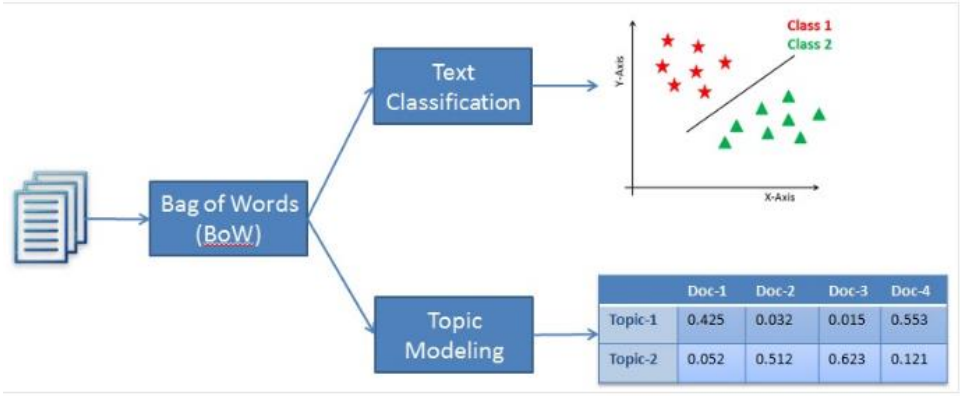

Рисунок 4.2 – Порівняння між класифікацією тексту та тематичним моделюванням

ЛСА (латентний семантичний аналіз), також відомий як ЛСІ (латентний семантичний індекс). ЛСА використовує модель пакету слів (BoW), яка призводить до матриці термінів-документів (поява термінів у документі). Рядки представляють терміни, а стовпці представляють документи. ЛСА вивчає приховані теми, виконуючи декомпозицію матриці на матриці термінів документа за допомогою декомпозиції за сингулярним значенням. ЛСА зазвичай використовується як техніка зменшення розмірів або шуму [5].

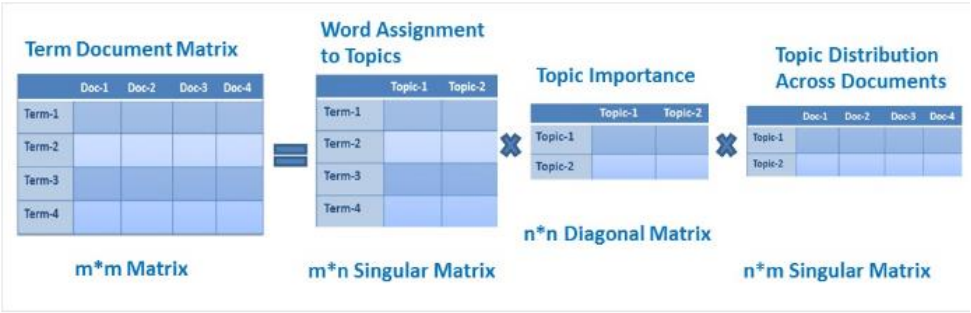

Рисунок 4.3 – Матриці латентно-семантичного аналізу

Розкладання сингулярного значення — це метод факторизації матриці, який представляє матрицю у вигляді добутку двох матриць. Він пропонує різні корисні програми в обробці сигналів, психології, соціології, науці про клімат і атмосферу, статистиці та астрономії.

$$M \sum V^*$$

де M є матрицею m×m;

U — ліва сингулярна матриця m×n;

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

V — 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.

Починаємо побудову програми та перш за все імпортуємо потрібні gensim та nltk бібліотеки [18].

Спочатку створимо функцію завантаження даних:

```
def load_data(path,file_name):
    """
    Input  : path and file_name
    Purpose: loading text file
    Output : list of paragraphs/documents and
             title(initial 100 words considred as title of document)
    """
    documents_list = []
    titles=[]
    with open( os.path.join(path, file_name) ,"r") as fin:
        for line in fin.readlines():
            text = line.strip()
            documents_list.append(text)
    print("Total Number of Documents:",len(documents_list))
    titles.append( text[0:min(len(text),100)] )
    return documents_list,titles
```

Після функції завантаження даних потрібно попередньо обробити текст. Для попередньої обробки тексту виконуються такі дії:

– Токенізуємо текстові статті;

– Видаляємо стоп-слова;

– Робимо стемінг тексту статті.

```
def preprocess_data(doc_set):
    """
    Input  : docuemnt list
    Purpose: preprocess text (tokenize, removing stopwords, and stemming)
    Output : preprocessed text
    """
    # initialize regex tokenizer
    tokenizer = RegexpTokenizer(r'\w+')
    # create English stop words list
    en_stop = set(stopwords.words('english'))
    # Create p_stemmer of class PorterStemmer
    p_stemmer = PorterStemmer()
    # list for tokenized documents in loop
    texts = []
    # loop through document list
    for i in doc_set:
        # clean and tokenize document string
        raw = i.lower()
        tokens = tokenizer.tokenize(raw)
        # remove stop words from tokens
        stopped_tokens = [i for i in tokens if not i in en_stop]
        # stem tokens
        stemmed_tokens = [p_stemmer.stem(i) for i in stopped_tokens]
        # add tokens to list
        texts.append(stemmed_tokens)
    return texts
```

Наступний крок — підготовка корпусу. Тут потрібно створити матрицю термінів-документів і словник термінів.

```
def prepare_corpus(doc_clean):
    """
    Input  : clean document
    Purpose: create term dictionary of our courpus and Converting list of do
cuments (corpus) into Document Term Matrix
    Output : term dictionary and Document Term Matrix
    """
    # Creating the term dictionary of our courpus, where every unique term i
s assigned an index. dictionary = corpora.Dictionary(doc_clean)
    dictionary = corpora.Dictionary(doc_clean)
    # Converting list of documents (corpus) into Document Term Matrix using
dictionary prepared above.
    doc_term_matrix = [dictionary.doc2bow(doc) for doc in doc_clean]
    # generate LDA model
    return dictionary,doc_term_matrix
```

Після створення корпусу можна створити модель за допомогою ЛСА.

```
def create_gensim_lsa_model(doc_clean,number_of_topics,words):
    """
    Input  : clean document, number of topics and number of words associated
 with each topic
    Purpose: create LSA model using gensim
    Output : return LSA model
    """
    dictionary,doc_term_matrix=prepare_corpus(doc_clean)
    # generate LSA model
    lsamodel = LsiModel(doc_term_matrix, num_topics=number_of_topics, id2wor
d = dictionary)  # train model
    print(lsamodel.print_topics(num_topics=number_of_topics, num_words=words
))
    return lsamodel
```

Необхідно зробити ще один додатковий крок, щоб оптимізувати результати шляхом визначення оптимальної кількості тем. Генеруємо оцінки узгодженості, щоб визначити оптимальну кількість тем.

```
def compute_coherence_values(dictionary, doc_term_matrix, doc_clean, stop, s
tart=2, step=3):
    """
    Input   : dictionary : Gensim dictionary
              corpus : Gensim corpus
              texts : List of input texts
              stop : Max num of topics
    purpose : Compute c_v coherence for various number of topics
    Output  : model_list : List of LSA topic models
              coherence_values : Coherence values corresponding to the LDA m
odel with respective number of topics
    """
    coherence_values = []
    model_list = []
    for num_topics in range(start, stop, step):
        # generate LSA model
        model = LsiModel(doc_term_matrix, num_topics=number_of_topics, id2wo
rd = dictionary)  # train model
        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=doc_clean, dictio
nary=dictionary, coherence='c_v')
        coherence_values.append(coherencemodel.get_coherence())
    return model_list, coherence_values
```

Тепер побудуємо значення оцінки когерентності.

```
# LSA Model
number_of_topics=7
words=10
```

```
document_list,titles=load_data("","The Guardian.txt")
clean_text=preprocess_data(document_list)
model=create_gensim_lsa_model(clean_text,number_of_topics,words)
```

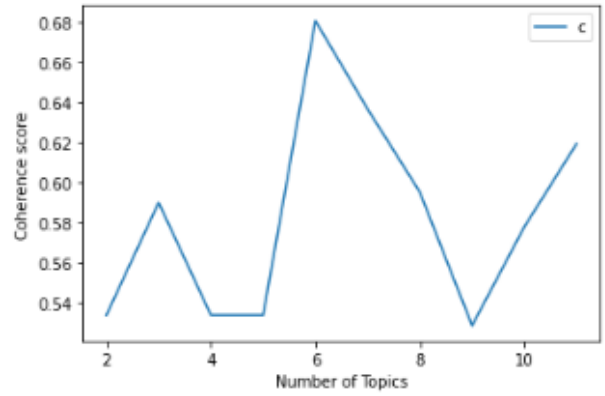

Рисунок 4.4 – Графік коефіцієнту подібності статті The Guardians до статей в Інтернеті

По графіку можна зробити висновки, що максимальний коефіцієнт подібності у статті це 0,68 з 6 знайденою статтею. Всього програма знайшла 18 подібних статей за даною темою.

Алгоритм ЛСА є найпростішим методом, який легко зрозуміти та реалізувати. Він також пропонує кращі результати порівняно з моделлю векторного простору. Він швидший порівняно з іншими доступними алгоритмами, оскільки включає лише декомпозицію матриці термінів документа.

Розмір латентної теми залежить від рангу матриці, тому ми не можемо розширити це обмеження. Розкладена матриця ЛСА є дуже щільною матрицею, тому важко індексувати окремий розмір. ЛСА не в змозі охопити кілька значень слів. Це не легше реалізувати порівняно з ЛДА (прихований розподіл Дірхле). Він забезпечує нижчу точність, ніж ЛДА.

4.4 Метод адаптивних онтологій(концептуальних графів)

Першим кроком для опрацювання даного методу побудуємо спільну онтологію для 2 статей про аналіз руйнувань під час війни в Україні. Архітектуру плагінів Protégé можна адаптувати для створення як простих, так і складних програм на основі онтології. Розробники можуть інтегрувати вихідні дані Protégé із системами правил або іншими засобами вирішення проблем для створення широкого спектру інтелектуальних систем.

Protégé повністю підтримує останню мову веб-онтології OWL 2 і специфікації RDF від World Wide Web Consortium.

Protégé базується на Java, є розширюваним і забезпечує середовище plug-and-play, що робить його гнучкою основою для швидкого створення прототипів і розробки додатків [5].

В OWL основним класом, на основі якого створюються класи онтології, є клас "owl:Class". Інші класи по відношенню до нього вважаються дочірніми підкласами. Класи одного рівня ієрархії в Protégé називаються Sibling Class.

На вкладці «Використання» для класу, вибраного в ієрархії, відображаються його зв’язки з батьківським і дочірнім класами, його властивості, екземпляри тощо.

У правій нижній панелі «Опис» можна вказати додаткові характеристики класу. Наприклад, його еквівалентність іншим класам або неможливість приналежності екземплярів класу іншим класам є забороною множинного успадкування.

Властивості класів та їх екземплярів (предикатів RDF-трійок) діляться на два типи:

– властивості зв'язку задаються на вкладці «Властивості об'єкта» і визначають деякі зв'язки між двома індивідами (примірниками класів), тобто суб'єктом і об'єктом трійки RDF будуть індивіди;

– властивості даних задаються на вкладці «Властивості даних» і визначають деякі фактичні характеристики індивідів (екземплярів класів), тобто суб'єктом трійки RDF буде індивід, а об'єктом – значення ознаки у формі терміну, числа, дати тощо.

Створення та редагування пов'язаних властивостей здійснюється на вкладці «Властивості об'єкта».

Автоматизована побудова та виведення графа онтології та взаємозв'язків між класами можна виконати за допомогою функцій «OWLViz» і «OntoGraf».

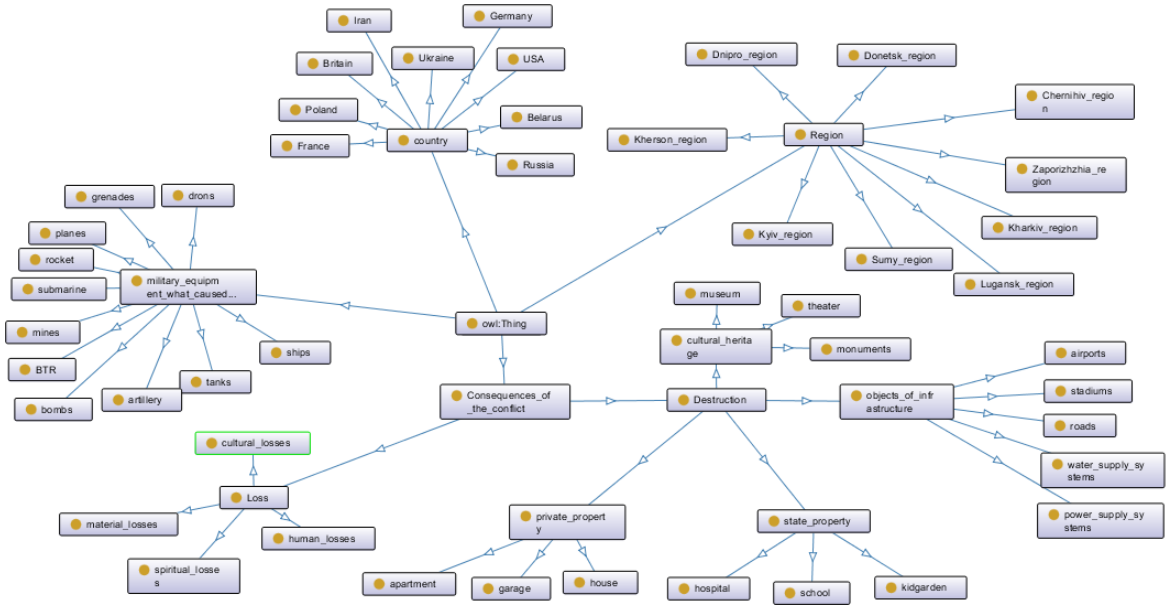

Рисунок 4.5 – Онтологія методів оцінки руйнувань внаслідок війни.

Після побудови онтології будуємо концептуальний граф по 1 статті. Для проведення розрахунків візьмемо за основу анотації до статей. Нижче будуть приведені анотації які будуть опрацьовані даним методом.

1. «The grinding conflict in Ukraine has rocked the global order, wrought destruction that "defies comprehension" and shows little sign of ending soon.» - NYT.

2. «Kyiv School of Economics estimates cost of conflict could rise to $600bn, almost four times the nation's GDP.» - The Guardian.

Будуємо концептуальний граф до першої анотації. Вага зв'язків була отримана із побудованої онтології.

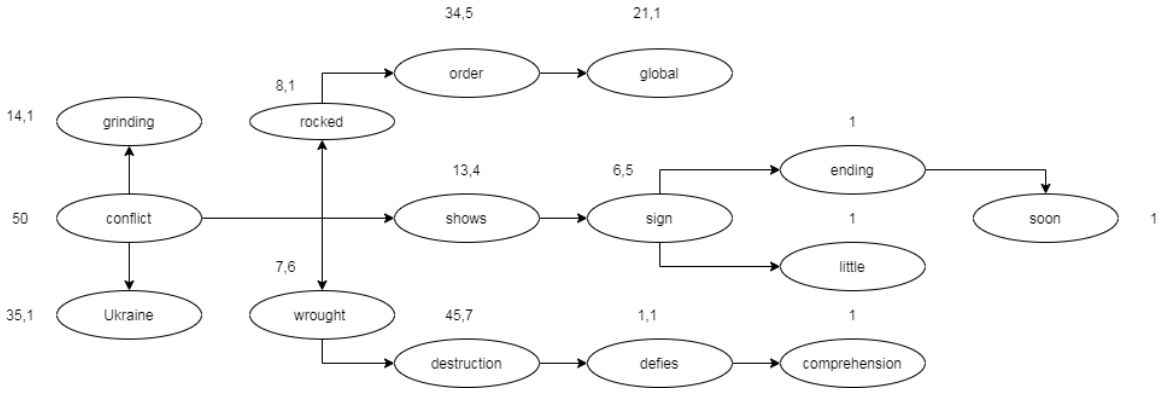

Рисунок 4.6 – Концептуальний граф 1-ї анотації

Тепер будуємо концептуальний граф до другої анотації.

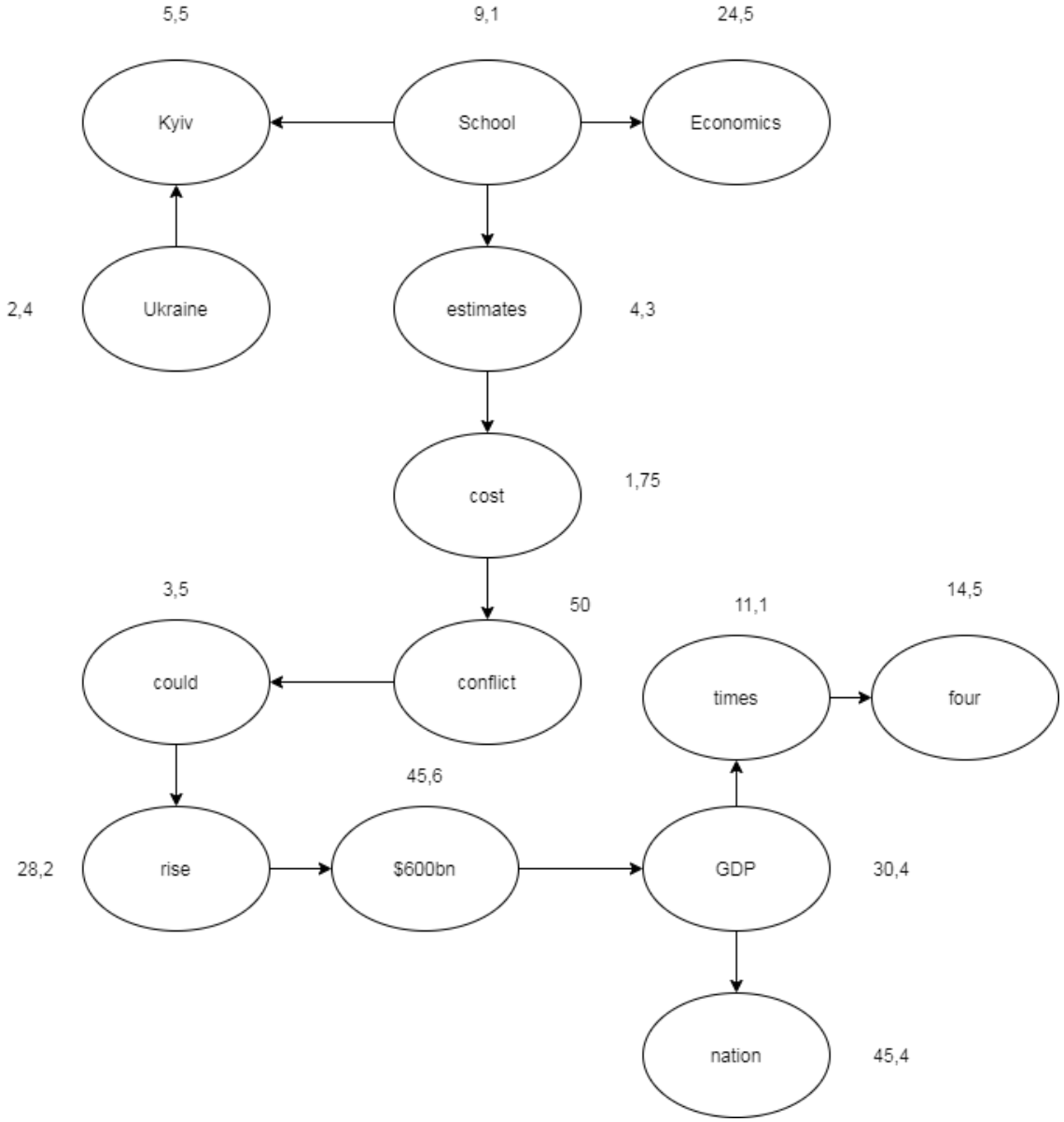

Рисунок 4.7 – Концептуальний граф 2-ї анотації

У випадку коли в концептуальному графі відсутній зв’язок, то вагу такого зв’язку прирівнювали до 5, що є середнім значенням ваги зв’язків.

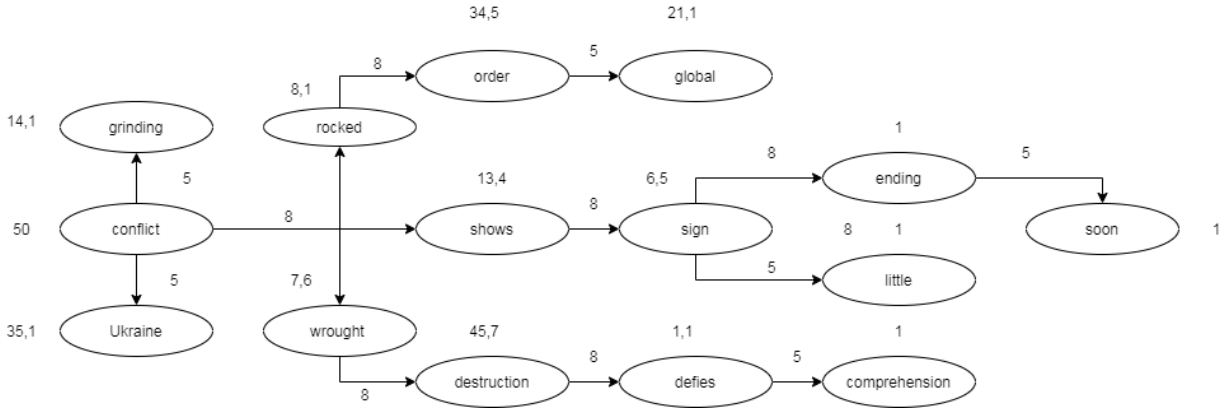

Рисунок 4.8 – Доповнений концептуальний граф 1-ї анотації

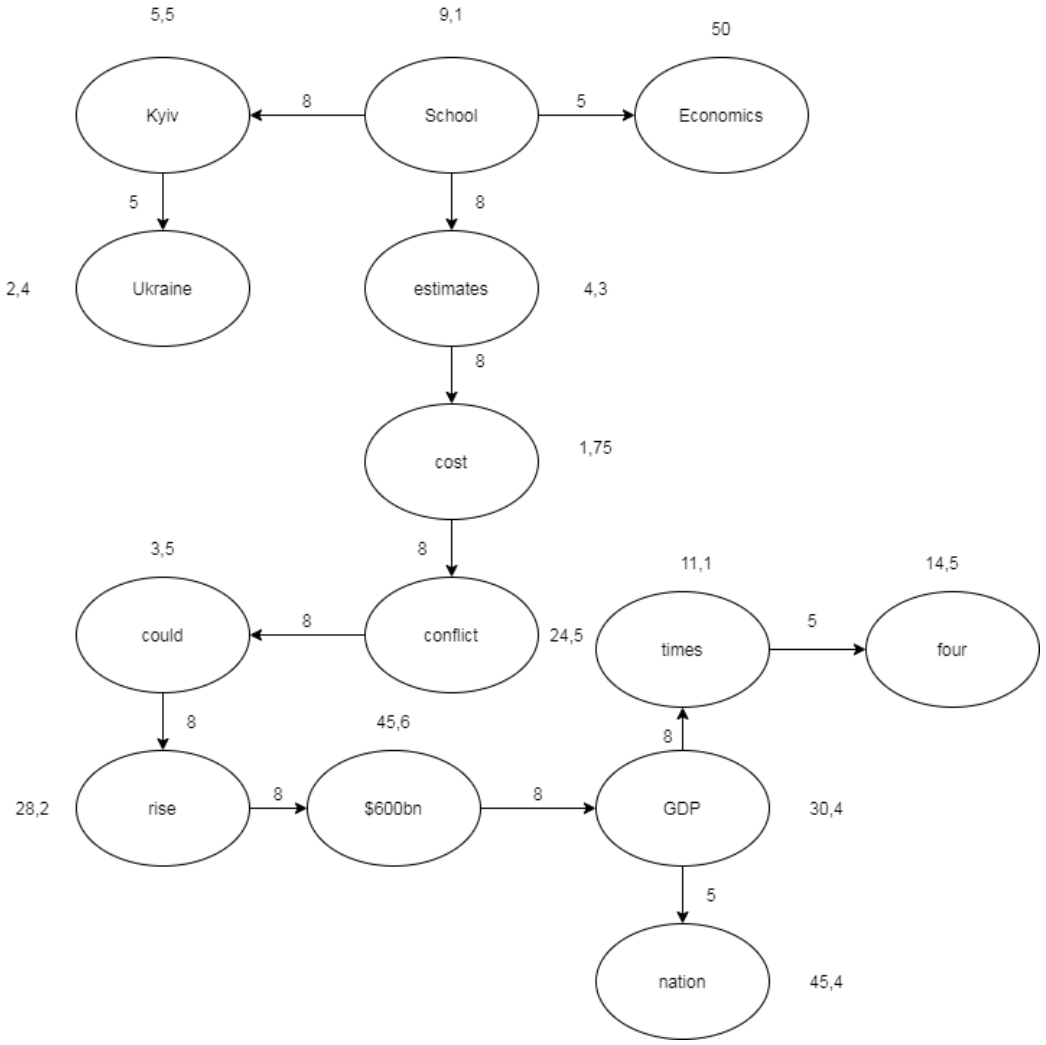

Рисунок 4.9 – Доповнений концептуальний граф 2-ї анотації

Відстань між двома вершинами графа C_i та C_j, якщо ці вершини з'єднані дугою, визначимо як:

$$d_{ij} = \frac{Q}{L_{ij}(W_i + W_j)}$$

де $L_{ij}(W_i + W_j)$ – сила зв'язку між вершинами графа C_i та C_j;

Q – константа, яка залежить від конкретної онтології.

Для даного випадку Q = 100. Сила зв'язку між вершинами обернено пропорційна відстані між ними [7].

Позначимо через d_{ij}^*– найкоротший шлях між вершинами C_i та C_j,. Її можна обчислити за допомогою відомих алгоритмів, наприклад Форда, Дейкстри, Флойда-Уоршалла.

$$d_{1,2} = \frac{100}{5(50 + 35{,}1)} = 0{,}24 \qquad d_{6,14} = \frac{100}{8(7{,}6 + 45{,}7)} = 0{,}24$$

$$d_{1,3} = \frac{100}{5(14{,}1 + 50)} = 0{,}31 \qquad d_{11,12} = \frac{100}{8(1{,}1 + 45{,}7)} = 0{,}27$$

$$d_{1,4} = \frac{100}{8(50 + 13{,}4)} = 0{,}2 \qquad d_{12,13} = \frac{100}{5(1{,}1 + 1)} = 9{,}52$$

$$d_{1,5} = \frac{100}{8(8{,}1 + 50)} = 0{,}22 \qquad d_{11,12} = \frac{100}{8(13{,}4 + 6{,}5)} = 0{,}63$$

$$d_{1,6} = \frac{100}{8(50 + 7{,}6)} = 0{,}22 \qquad d_{9,10} = \frac{100}{8(6{,}5 + 1)} = 1{,}7$$

$$d_{5,7} = \frac{100}{8(34{,}5 + 8{,}1)} = 0{,}29 \qquad d_{9,15} = \frac{100}{5(6{,}5 + 1)} = 2{,}7$$

$$d_{7,8} = \frac{100}{5(34{,}5 + 21{,}1)} = 0{,}36 \qquad d_{10,14} = \frac{100}{5(1 + 1)} = 10$$

Середня відстань для вершини C_i обчислюється згідно з формулою:

$$\bar{d}_i = \frac{\sum_{j=1, j\neq i}^{n} d_{ij}^*}{n-1}$$

де n – кількість вершин графа.

Рохрахуємо та отримуємо:

$\bar{d}_1$=0,24
$\bar{d}_2$=0,24
$\bar{d}_3$=0,24
$\bar{d}_4$=0,42
$\bar{d}_5$=0,26
$\bar{d}_6$=0,23
$\bar{d}_7$=0,33
$\bar{d}_8$=0,36
$\bar{d}_9$=1,68
$\bar{d}_{10}$=5,9
$\bar{d}_{11}$=0,26
$\bar{d}_{12}$=4,9
$\bar{d}_{13}$=9,52
$\bar{d}_{14}$=10
$\bar{d}_{15}$=2,7

Отримаємо наступний зважений граф для 1 анотації статі NYT:

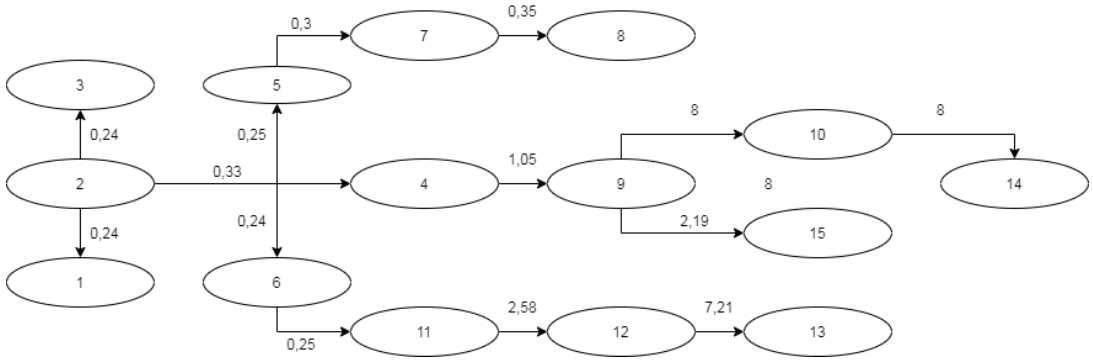

Рисунок 4.10 – Зважений концептуальний граф статті NYT

Таким же чином проведемо розрахунки для статті The Guardian:

$$d_{1,2} = \frac{100}{5(5{,}5 + 2{,}4)} = 2{,}53$$

$$d_{2,3} = \frac{100}{8(9{,}1 + 5{,}5)} = 0{,}86$$

$$d_{8,9} = \frac{100}{8(3{,}5 + 28{,}2)} = 0{,}39$$

$$d_{9,10} = \frac{100}{8(28{,}2 + 45{,}6)} = 0{,}17$$

$$d_{3,4} = \frac{100}{5(50+9,1)} = 0,34$$

$$d_{3,5} = \frac{100}{8(9,1+4,3)} = 0,93$$

$$d_{5,6} = \frac{100}{8(4,3+1,75)} = 2,07$$

$$d_{6,7} = \frac{100}{8(24,5+1,75)} = 0,49$$

$$d_{7,8} = \frac{100}{8(3,5+24,5)} = 0,45$$

$$d_{10,11} = \frac{100}{8(45,6+30,4)} = 0,17$$

$$d_{11,12} = \frac{100}{5(30,4+45,4)} = 0,21$$

$$d_{11,13} = \frac{100}{8(30,4+11,1)} = 0,3$$

$$d_{13,14} = \frac{100}{5(14,5+11,1)} = 0,78$$

Далі розраховуємо середню відстань та отримуємо:

$\bar{d}_1$=2,53

$\bar{d}_2$=1,7

$\bar{d}_3$=0,71

$\bar{d}_4$=0,34

$\bar{d}_5$=1,5

$\bar{d}_6$=1,28

$\bar{d}_7$=0,47

$\bar{d}_8$=0,42

$\bar{d}_9$=0,28

$\bar{d}_{10}$=0,17

$\bar{d}_{11}$=0,23

$\bar{d}_{12}$=0,21

$\bar{d}_{13}$=0,54

$\bar{d}_{14}$=0,78

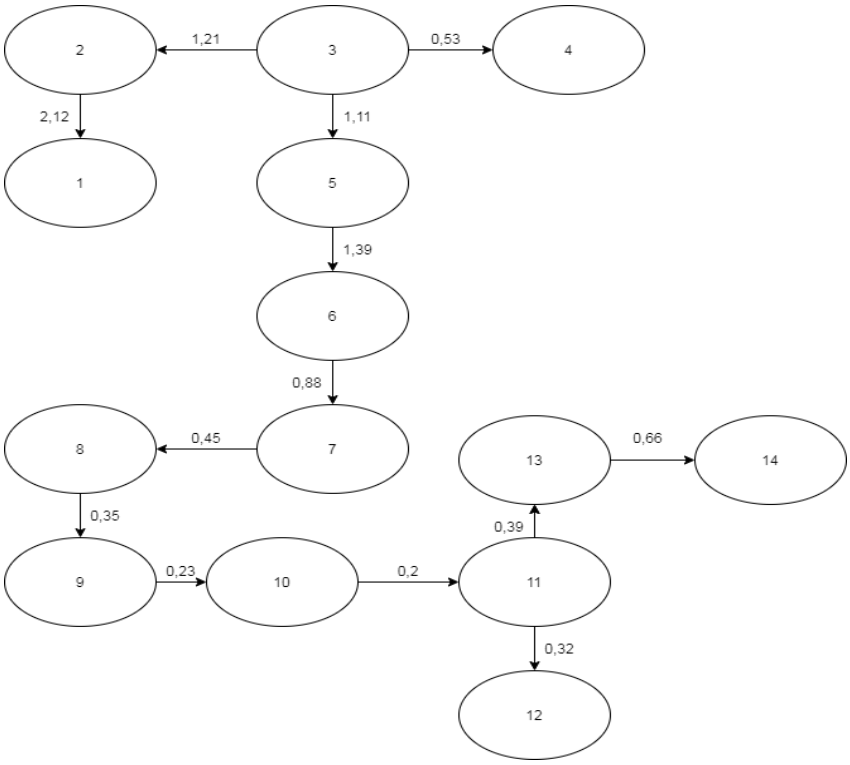

Рисунок 4.11 – Зважений концептуальний граф 2 статті

Накладаємо обидва графи. У випадку, якщо вони матимуть спільні дуги, то відстань між вершинами, що з’єднані такими дугами, визначається як середня відстань двох графів:

$$\bar{d}_{1,2} = \frac{\bar{d}_1 + \bar{d}_2}{2}$$

$\bar{d}_{1,2}$=2,12

$\bar{d}_{2,3}$=1,21

$\bar{d}_{3,4}$=0,53

$\bar{d}_{3,5}$=1,11

$\bar{d}_{5,6}$=1,39

$\bar{d}_{6,7}$=0,88

$\bar{d}_{7,8}$=0,45

$\bar{d}_{8,9}$=0,35

$\bar{d}_{9,10}$=0,23

$\bar{d}_{10,11}$=0,2

$\bar{d}_{11,12}$=0,22

$\bar{d}_{11,13}$=0,39

$\bar{d}_{13,14}$=0,66

Перша та друга статті можна зв’язати за допомогою вершини conflict (друга в першій статті і сьома в другій статті). Тоді $d_2^{1*} = 6,71$ та $d_7^{2*} = 0,24$. А відстань між першою та другою статтями дорівнює 6,95. Тобто $\bar{d}^{12} = 6,71 + 0,24 = 6,95$.

Використання для порівняння документів методу адаптивних онтологій дозволяє:

– Проводити порівняння текстів незважаючи на їх розмір;

– Виконувати порівняння різних фрагментів одного й того ж тексту, що можуть перетинатися у якійсь частині;

– Дає можливість оцінити релевантність досліджуваного тексту до заданої бази знань, представленої концептуальним графом.

Окрім того, з цілю пришвидшення операцій по порівнянню текстів метод допускає їх скорочення. Скорочення проводиться шляхом відкидання тих вузлів концептуального графа, які мають найменші коефіцієнти важливості у предметній області. При скороченні центри семантичної ваги не змінюються, що забезпечує незалежність проведеної оцінки подібності текстів за змістом та від розміру [7].

На основі методу є можливість здійснювати автоматизований пошук документів, які найбільше відповідають запиту-взірцеві в мережі Інтернет, а також проводити інтелектуальний аналіз тексту, його класифікацію та ранжування за релевантністю до заданої предметної області [5].

4.5 Порівняння методів оцінювання подібності статей

Після дослідження методів оцінювання подібності документів та проведення розрахунків за кожним з них можна підвести висновки. Найпростішими та найшвидшими в реалізації є методи по коефіцієнту Жаккара

та по коефіцієнту Дайса. При порівнянні отриманих результатів можна зробити висновки, що при підрахунку за методом по коефіцієнту Дайса при меншій кількості документів для порівняння результат буде більш точним, чим за методом по коефіцієнту Жаккара, але у випадку зростання кількості порівнюваних документів коефіцієнт Жаккара видасть більш точний результат.

У випадку з методом латентно-семантичного індексування ситуація більш складна. З основних мінусів даного методу необхідно виділити те, що опрацьовувати його вручну, без застосування програми, або як у даній роботі розробки власного програмного застосунку, складно реалізована задача. Також водночас як і мінус так і плюс цього методу є те, що він працює зі своєю бібліотекою статей з мережі Інтернет. Тому неможливо порівняти за допомогою нього лише 2 файли. Але метод високоточний та його можна застосовувати у великій кількості областей.

Метод зважених концептуальних графів можна вважати найкращим з усіх досліджених методів. Він дає можливість проводити порівняння текстів незважаючи на їх розмір, виконувати порівняння різних фрагментів одного й того ж тексту, що можуть перетинатися у якійсь частині та дає можливість оцінити релевантність досліджуваного тексту до заданої бази знань, представленої концептуальним графом. З мінусів можна виділити важкість ручного підрахунку, порівняно з іншими досліджуваними методами, в яких також проводився ручний підрахунок, а не за допомогою програми. Також у випадку реалізації даного методу у вигляді програмного застосунку при високій точності підрахунку область застосування буде обмежуватися предметною областю побудованої онтологією. Тому для програмної реалізації було обрано метод за коефіцієнтом Дайса.

5 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Обґрунтування вибору середовища та мови реалізації

У світі програмування Python відомий як одна з найпопулярніших і швидкозростаючих мов програмування. Її можна визначити як високорівневу, інтерпретовану, об’єктно-орієнтовану мову сценаріїв і мову загального призначення.

Python — це мова комп’ютерного програмування високого рівня з динамічною семантикою. Це високо інтерпретована та об’єктно-орієнтована мова з вбудованими структурами даних у поєднанні з динамічною типізацією та динамічним зв’язуванням, що робить її дуже привабливою для швидкої розробки додатків.

Python використовується у всьому, від машинного навчання до створення веб-сайтів і тестування програмного забезпечення. Звичайно, мова Python найчастіше використовується для створення мобільних, веб- та десктопних програм. Під час тестування цих програм також можна використовувати Python для створення алгоритмів рекомендацій, розробки програмного забезпечення, яке керує безпілотними автомобілями, автоматизує повсякденні завдання, від веб-розробки до сценаріїв, тестування програмного забезпечення та створення прототипів.

Python має широкий вибір бібліотек і фреймворків, що є однією з найбільших переваг Python. Від NumPy до TensorFlow. Бібліотека Python використовується для всього, починаючи від візуалізації даних, машинного навчання, науки про дані, обробки природної мови та складного аналізу даних. Наявність великої бібліотеки з керуванням пам’яттю та порожнім дизайном допомагає підвищити продуктивність розробника Python. В результаті розробники можуть керувати базами даних, документацією, веб-браузерами; виконувати модульне тестування та багато інших функцій. Крім того, Python можна використовувати для розробки багатьох завдань, таких як розробка веб-

додатків і програм для настільних комп’ютерів, розробка складних систем обчислення, систем керування життєзабезпеченням, ігор тощо [18].

Google Colaboratory або Colab, це ще один хмарний сервіс від Google Research. Це IDE, яка дозволяє будь-якому користувачеві написати вихідний код у редакторі та запустити його з браузера. Зокрема, Colab підтримує мову програмування Python і орієнтований на завдання машинного навчання, аналіз даних, освітні проекти тощо.

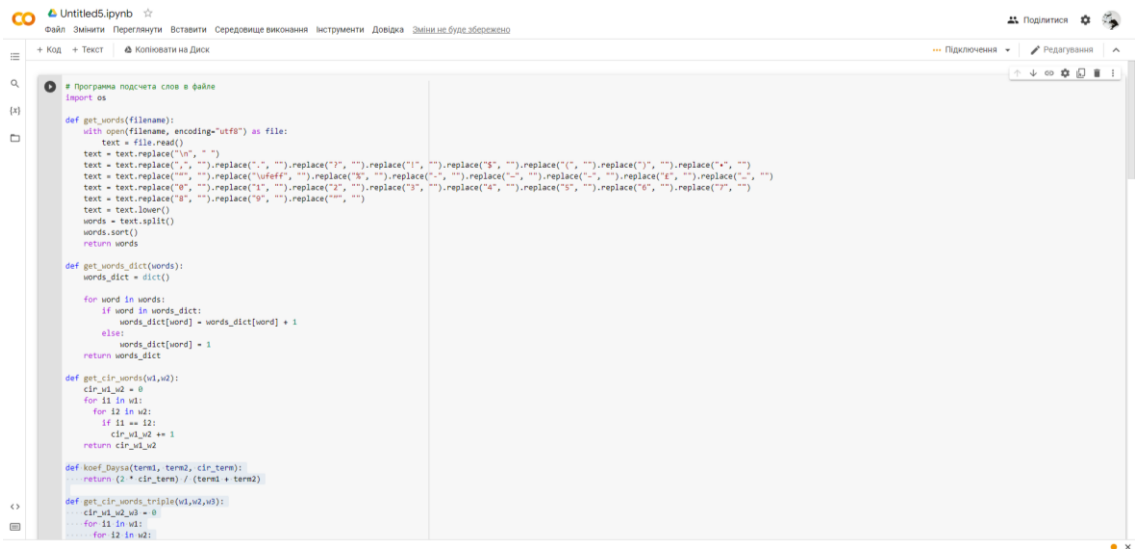

Рисунок 5.1 – Робочий простір у середовищі Google Colaboratory

Ця послуга, заснована на Jupyter Notebook, абсолютно безкоштовно розміщена, не вимагає налаштувань, а також не потрібно буде завантажувати або встановлювати Jupyter. Colab пропонує користувачеві вичерпні ресурси для редагування та тестування коду.

5.2 Опис програмного застосунку

Для реалізації даного програмного забезпечення було обрано мову програмування Python та як середовище Colab. З потрібного функціоналу, для вирішення поставленої задачі необхідно:

– Підрахунок загальної кількості слів у статті;
– Підрахунок термінів у документі;
– Можливість додавання електронних документів користувача;
– Підрахунок за методом по коефіцієнту Дайса подібності статей.

```
def get_words(filename):
    with open(filename, encoding="utf8") as file:
        text = file.read()
    text = text.replace("\n", " ")
    text = text.replace(",", "").replace(".", "").replace("?", "").replace("
!", "").replace("$", "").replace("(", "").replace(")", "").replace("•", "")
    text = text.replace("^", "").replace("\ufeff", "").replace("%", "").repl
ace("-", "").replace("—", "").replace("-
", "").replace("£", "").replace("…", "")
    text = text.replace("0", "").replace("1", "").replace("2", "").replace("
3", "").replace("4", "").replace("5", "").replace("6", "").replace("7", "")
    text = text.replace("8", "").replace("9", "").replace("″", "")
    text = text.lower()
    words = text.split()
    words.sort()
    return words
```

У функції get_words() виробляється початкова сегментація тексту на слова. При цьому всі пунктуаційні знаки видаляються, а стоп слова замінюються на пробілі. Потім відбувається розбиття тексту на слова. У якості розділювача за умовчанням застосовується пробіл.

```
     def get_words_dict(words):
    words_dict = dict()
for word in words:
        if word in words_dict:
            words_dict[word] = words_dict[word] + 1
        else:
            words_dict[word] = 1
    return words_dict

def get_cir_words(w1,w2):
    cir_w1_w2 = 0
    for i1 in w1:
      for i2 in w2:
        if i1 == i2:
          cir_w1_w2 += 1
    return cir_w1_w2
```

Далі в функціях get_words_dict() отримуємо словник зі слів, де ключ - це унікальне слово, а значення - кількість введених даного слова в тексті.

```
def koef_Daysa(term1, term2, cir_term):
    return (2 * cir_term) / (term1 + term2)

def get_cir_words_triple(w1,w2,w3):
    cir_w1_w2_w3 = 0
    for i1 in w1:
      for i2 in w2:
        for i3 in w3:
          if i1 == i2 and i1 == i3 and i2 == i3:
            cir_w1_w2_w3 += 1
    return cir_w1_w2_w3

def koef_Daysa_triple(term1, term2, term3, cir_term):
    return (3 * cir_term) / (term1 + term2 + term3)

file_NYT = "NYT.txt"
file_Forbes = "Forbes.txt"
file_Guardian = "The Guardian.txt"


if not os.path.exists(file_NYT) and not os.path.exists(file_Forbes) and not
os.path.exists(file_Guardian):
    print("Якогось файлу не вистачає  ")
```

Після підрахунку загальної кількості слів та термінів необхідно розрахувати метод Дайса. Для цього формулу було адаптовано у функцію koef_Daysa та koef_Daysa_triple з можливістю підрахунку коефіцієнту подібності для 2-3 статтей. Далі йде перевірка існування потрібних файлів, розрахунки та виведення результату. За підсумком отримуємо результат зображений на рисунку 5.2

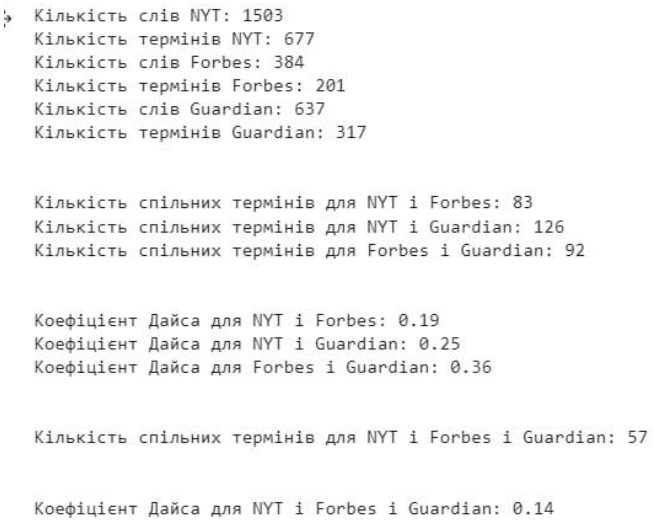

Рисунок 5.2 – Результат роботи програми

ВИСНОВКИ

В ході дипломної роботи виконано такі завдання:

– Розглянуто основні типи аналізу подібності – морфологічний, лексичний, семантичний та синтаксичний;

– проведено аналіз статей іноземних видань про руйнування під час війни в Україні;

– проведено розрахунки за 4 методами - латентно-семантичне індексування, метод зважених концептуальних графів, коефіцієнт Жаккара, коефіцієнт Дайса;

– для методу концептуальних графів побудовано онтологію;

– для методів латентно-семантичного індексування та за коефіцієнтом Дайса розроблено програми на мові Python в середовищі Colab.

З можливих варіантів покращення роботи можна виділити такі варіанти як:

– Розглянути інші методи пошуку подібності в документах;

– розробка зручного інтерфейсу для програм;

– розширення онтології додаванням нових класів та підкласів.

ПЕРЕЛІК ПОСИЛАНЬ

1. **Стрехль, А.** Вплив заходів подібності на кластеризацію веб-сторінок / А. Стрехль, Д. Гот, Р. Мучні // Семінар зі штучного інтелекту для веб-пошуку. – 2000. – №5. – С. 58–64.

2. **Уайт, Р. В.** Дослідження заходів подібності тем / Р. В. Уайт, Д. М. Хосе // Журнал досліджень штучного інтелекту. – 2004. – №27. – С. 520–521.

3. **Хуанг , А.** Міри подібності для кластеризації текстових документів / А. Хуанг // Матеріали шостої студентської конференції з досліджень інформатики в Новій Зеландії. – 2008. – №4. – С. 49–56.

4. **Форсайт, А. Р.** Несхожість документів у межах і між мовами: порівняльне дослідження / А. Р. Форсайт, В. С. Шароф // Літературно-лінгвістичний комп’ютер. – 2014. – №29. – С. 6–22.

5. **Литвин, В. В.** Методи та засоби опрацювання інформаційних ресурсів на основі онтологій: Монографія / В. В. Литвин, В. А. Висоцька, Д. Г. Досин. – Львів: Видавництво Національного університету «Львівська політехніка», 2016. – 460 с

6. **Даревич, Р. Р.** Оцінка подібності текстових документів на основі визначення інформаційної ваги елементів бази знань / Р. Р.Даревич, Д. Г. Досин, В. В. Литвин, З. Т. Назарчук // Штучний інтелект. – 2006. – №3. – С. 500–509.

7. **Литвин, В. В.** Метод оцінювання подібності документів, доповнених контекстом з онтології / В. В. Литвин. // Штучний інтелект. – 2008. – №18. – С. 6–22.

8. **Даревич, Р.Р.** Підвищення ефективності інтелектуального аналізу тексту шляхом зважування понять в моделі онтології // Штучний інтелект. – 2005. – № 3. – С. 571-577.

9. **Онищенко, К.** Аналіз Методів Обробки Природної Мови / К. Онищенко, Я. Данієль, Р. Каменєв. // Інформаційні системи та технології. – 2020. – №14. – С. 186–190.

10. **Дарчук, Н. П.** Комп’ютерна лінгвістика (автоматичне опрацювання тексту): підручник / Н. П. Дарчук. – К.: Видавничо-поліграфічний центр “Київський університет”, 2008. – 351 с.

11. **Тарануха, В. Ю.** Інтелектуальна обробка текстів [Електронний ресурс] : навчальний посібник / В. Ю. Тарануха. – К., 2014. – 80 с. – Режим доступу http://www.cyb.univ.kiev.ua/library/books/taranukha-40.pdf

12. 100 днів війни: смерть, руйнування та втрати [Електронний ресурс] // New York Times. – 2022. – Режим доступу: https://www.nytimes.com/2022/06/03/world/europe/russia-ukraine-war-100-days.html.

13. Війна завдала шкоди інфраструктурі України на 108 мільярдів доларів, – дослідження [Електронний ресурс] // Forbes. – 2022. – Режим доступу: https://www.forbes.com/sites/madelinehalpert/2022/08/02/war-has-caused-108-billion-in-damage-to-ukraines-infrastructure-study-finds/?sh=6f491abc23e5.

14. Війна Росії в Україні спричиняє 36 мільярдів будівель збитків на тиждень [Електронний ресурс] // The Guardian. – 2022. – Режим доступу: https://www.theguardian.com/world/2022/may/03/russias-war-in-ukraine-causing-36bn-of- building-damage-a-week.

15. Вторгнення Росії завдає шкоди всій Україні [Електронний ресурс] // Reuters. – 2022. – Режим доступу: https://www.reuters.com/graphics/UKRAINE-CRISIS/DAMAGE/lbpgnqnljvq/.

16. Україна: Цикл смерті, руйнування, дислокації та підриву «має припинитися» [Електронний ресурс] // UN news. – 2022. – Режим: https://news.un.org/en/story/2022/06/1121592.

17. Війна Росії в Україні: огляди RAND [Електронний ресурс] // RAND. – 2022. – Режим доступу: https://www.rand.org/latest/russia-ukraine.html.

18. **Мюллер, А**. Введення в машинне навчання за допомогою Python / А. Мюллер, С. Гвидо. – Київ: Диалектика, 2017. – 190 с.

ДОДАТОК А
Код програми за методом латентно-семантичного індексування

```
from nltk.corpus.reader import nltk
#import modules
import os.path
import nltk
nltk.download('stopwords')
from gensim import corpora
from gensim.models import LsiModel
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from gensim.models.coherencemodel import CoherenceModel
import matplotlib.pyplot as plt
def load_data(path,file_name):
    """
    Input  : path and file_name
    Purpose: loading text file
    Output : list of paragraphs/documents and
             title(initial 100 words considred as title of document)
    """
    documents_list = []
    titles=[]
    with open( os.path.join(path, file_name) ,"r") as fin:
        for line in fin.readlines():
            text = line.strip()
            documents_list.append(text)
    print("Total Number of Documents:",len(documents_list))
    titles.append( text[0:min(len(text),100)] )
    return documents_list,titles
def preprocess_data(doc_set):
    """
    Input  : docuemnt list
    Purpose: preprocess text (tokenize, removing stopwords, and stemming)
    Output : preprocessed text
    """
    # initialize regex tokenizer
    tokenizer = RegexpTokenizer(r'\w+')
    # create English stop words list
    en_stop = set(stopwords.words('english'))
    # Create p_stemmer of class PorterStemmer
    p_stemmer = PorterStemmer()
    # list for tokenized documents in loop
    texts = []
    # loop through document list
    for i in doc_set:
        # clean and tokenize document string
        raw = i.lower()
```

```
        tokens = tokenizer.tokenize(raw)
        # remove stop words from tokens
        stopped_tokens = [i for i in tokens if not i in en_stop]
        # stem tokens
        stemmed_tokens = [p_stemmer.stem(i) for i in stopped_tokens]
        # add tokens to list
        texts.append(stemmed_tokens)
    return texts
def prepare_corpus(doc_clean):
    """
    Input  : clean document
    Purpose: create term dictionary of our courpus and Converting list of documents (co
rpus) into Document Term Matrix
    Output : term dictionary and Document Term Matrix
    """
    # Creating the term dictionary of our courpus, where every unique term is assigned
an index. dictionary = corpora.Dictionary(doc_clean)
    dictionary = corpora.Dictionary(doc_clean)
    # Converting list of documents (corpus) into Document Term Matrix using dictionary
prepared above.
    doc_term_matrix = [dictionary.doc2bow(doc) for doc in doc_clean]
    # generate LDA model
    return dictionary,doc_term_matrix
def create_gensim_lsa_model(doc_clean,number_of_topics,words):
    """
    Input  : clean document, number of topics and number of words associated with each
topic
    Purpose: create LSA model using gensim
    Output : return LSA model
    """
    dictionary,doc_term_matrix=prepare_corpus(doc_clean)
    # generate LSA model
    lsamodel = LsiModel(doc_term_matrix, num_topics=number_of_topics, id2word = diction
ary)  # train model
    print(lsamodel.print_topics(num_topics=number_of_topics, num_words=words))
    return lsamodel
def compute_coherence_values(dictionary, doc_term_matrix, doc_clean, stop, start=2, ste
p=3):
    """
    Input   : dictionary : Gensim dictionary
              corpus : Gensim corpus
              texts : List of input texts
              stop : Max num of topics
    purpose : Compute c_v coherence for various number of topics
    Output  : model_list : List of LSA topic models
              coherence_values : Coherence values corresponding to the LDA model with r
espective number of topics
    """
    coherence_values = []
    model_list = []
```

```
    for num_topics in range(start, stop, step):
        # generate LSA model
        model = LsiModel(doc_term_matrix, num_topics=number_of_topics, id2word = dictio
nary)  # train model
        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=doc_clean, dictionary=dictio
nary, coherence='c_v')
        coherence_values.append(coherencemodel.get_coherence())
    return model_list, coherence_values
# LSA Model
number_of_topics=7
words=10
document_list,titles=load_data("","Forbes.txt")
clean_text=preprocess_data(document_list)
model=create_gensim_lsa_model(clean_text,number_of_topics,words)
def plot_graph(doc_clean,start, stop, step):
    dictionary,doc_term_matrix=prepare_corpus(doc_clean)
    model_list, coherence_values = compute_coherence_values(dictionary, doc_term_matrix
,doc_clean,
                                                            stop, start, step)
    # Show graph
    x = range(start, stop, step)
    plt.plot(x, coherence_values)
    plt.xlabel("Number of Topics")
    plt.ylabel("Coherence score")
    plt.legend(("coherence_values"), loc='best')
    plt.show()

start,stop,step=2,12,1
plot_graph(clean_text,start,stop,step)
# LSA Model
number_of_topics=7
words=10
document_list,titles=load_data("","The Guardian.txt")
clean_text=preprocess_data(document_list)
model=create_gensim_lsa_model(clean_text,number_of_topics,words)
# LSA Model
number_of_topics=7
words=10
document_list,titles=load_data("","NYT.txt")
clean_text=preprocess_data(document_list)
model=create_gensim_lsa_model(clean_text,number_of_topics,words)
```

ДОДАТОК Б
Код програми за методом Дайса

```
import os

def get_words(filename):
    with open(filename, encoding="utf8") as file:
        text = file.read()
    text = text.replace("\n", " ")
    text = text.replace(",", "").replace(".", "").replace("?", "").replace("!", "").rep
lace("$", "").replace("(", "").replace(")", "").replace("•", "")
    text = text.replace("“", "").replace("\ufeff", "").replace("%", "").replace("-
", "").replace("—", "").replace("-", "").replace("£", "").replace("…", "")
    text = text.replace("0", "").replace("1", "").replace("2", "").replace("3", "").rep
lace("4", "").replace("5", "").replace("6", "").replace("7", "")
    text = text.replace("8", "").replace("9", "").replace("”", "")
    text = text.lower()
    words = text.split()
    words.sort()
    return words

def get_words_dict(words):
    words_dict = dict()

    for word in words:
        if word in words_dict:
            words_dict[word] = words_dict[word] + 1
        else:
            words_dict[word] = 1
    return words_dict

def get_cir_words(w1,w2):
    cir_w1_w2 = 0
    for i1 in w1:
      for i2 in w2:
        if i1 == i2:
          cir_w1_w2 += 1
    return cir_w1_w2

def koef_Daysa(term1, term2, cir_term):
    return (2 * cir_term) / (term1 + term2)

def get_cir_words_triple(w1,w2,w3):
    cir_w1_w2_w3 = 0
    for i1 in w1:
      for i2 in w2:
        for i3 in w3:
          if i1 == i2 and i1 == i3 and i2 == i3:
            cir_w1_w2_w3 += 1
```

```
    return cir_w1_w2_w3

def koef_Daysa_triple(term1, term2, term3, cir_term):
    return (3 * cir_term) / (term1 + term2 + term3)

file_NYT = "NYT.txt"
file_Forbes = "Forbes.txt"
file_Guardian = "The Guardian.txt"


if not os.path.exists(file_NYT) and not os.path.exists(file_Forbes) and not os.path.exi
sts(file_Guardian):
    print("Якогось файлу не вистачає немає")
else:
    NYT_words = get_words(file_NYT)
    Forbes_words = get_words(file_Forbes)
    Guardian_words = get_words(file_Guardian)
    NYT_words_dict = get_words_dict(NYT_words)
    Forbes_words_dict = get_words_dict(Forbes_words)
    Guardian_words_dict = get_words_dict(Guardian_words)
    print(f"Кількість слів NYT: {len(NYT_words)}")
    print(f"Кількість термінів NYT: {len(NYT_words_dict)}")
    print(f"Кількість слів Forbes: {len(Forbes_words)}")
    print(f"Кількість термінів Forbes: {len(Forbes_words_dict)}")
    print(f"Кількість слів Guardian: {len(Guardian_words)}")
    print(f"Кількість термінів Guardian: {len(Guardian_words_dict)}\n\n")
    #print("Всі слова:")
    #for word in words_dict:
    # if word.isdigit() == False:
    #   print(word.ljust(20), words_dict[word])
    clear_words_NYT = []
    for word in NYT_words_dict:
      if word.isdecimal() == False:
        clear_words_NYT.append(word)

    clear_words_Forbes = []
    for word in Forbes_words_dict:
      if word.isdecimal() == False:
        clear_words_Forbes.append(word)

    clear_words_Guardian = []
    for word in Guardian_words_dict:
      if word.isdecimal() == False:
        clear_words_Guardian.append(word)

    cir_NYT_Forbes = get_cir_words(clear_words_NYT,clear_words_Forbes)
    cir_NYT_Guardian = get_cir_words(clear_words_NYT,clear_words_Guardian)
    cir_Forbes_Guardian = get_cir_words(clear_words_Forbes,clear_words_Guardian)

    print(f"Кількість спільних термінів для NYT і Forbes: {cir_NYT_Forbes}")
```

```
    print(f"Кількість спільних термінів для NYT і Guardian: {cir_NYT_Guardian}")
    print(f"Кількість спільних термінів для Forbes і Guardian: {cir_Forbes_Guardian}\n\
n")

    koef_Daysa_NYT_Forbes = koef_Daysa(len(NYT_words_dict), len(Forbes_words_dict), cir
_NYT_Forbes)
    koef_Daysa_NYT_Guardian = koef_Daysa(len(NYT_words_dict), len(Guardian_words_dict),
 cir_NYT_Guardian)
    koef_Daysa_Forbes_Guardian = koef_Daysa(len(Forbes_words_dict), len(Guardian_words_
dict), cir_Forbes_Guardian)

    print(f"Коефіцієнт Дайса для NYT і Forbes: {round(koef_Daysa_NYT_Forbes,2)}")
    print(f"Коефіцієнт Дайса для NYT і Guardian: {round(koef_Daysa_NYT_Guardian,2)}")
    print(f"Коефіцієнт Дайса для Forbes і Guardian: {round(koef_Daysa_Forbes_Guardian,2
)}\n\n")


    cir_NYT_Forbes_Guardian  = get_cir_words_triple(clear_words_NYT,clear_words_Forbes,
 clear_words_Guardian)
    print(f"Кількість спільних термінів для NYT і Forbes і Guardian: {cir_NYT_Forbes_Gu
ardian}\n\n")

    koef_Daysa_NYT_Forbes_Guardian = koef_Daysa_triple(len(NYT_words_dict), len(Forbes_
words_dict), len(Guardian_words_dict), cir_NYT_Forbes_Guardian)
    print(f"Коефіцієнт Дайса для NYT і Forbes і Guardian: {round(koef_Daysa_NYT_Forbes_
Guardian,2)}\n\n")

    print(clear_words_NYT)
    print(clear_words_Forbes)
    print(clear_words_Guardian)
```