Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи визначення текстової близькості фраз
(тема)

Виконав:

студент II курсу, групи СПм-20-2

Ляшова А.О.
(прізвище, ініціали)

Спеціальність

123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: проф. Фесенко Т.Г.
(посада, прізвище, ініціали)

Допускається до захисту

В.о. зав. кафедри ЕОМ _____ Волк М.О.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
В.о. зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Ляшовій Анастасії Олександрівні
(прізвище, ім'я, по батькові)

1. Тема роботи Методи визначення текстової близькості фраз

затверджена наказом по університету від " 24 " березня 2022 р. № 413 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 2022 р.

3. Вхідні дані до роботи вхідне повідомлення від користувача, база еталонних значень, алгоритм Дамерау-Левенштейна

4. Перелік питань, що потрібно опрацювати у роботі
аналіз вхідних речень;
аналіз та обробка окремого слова;
розробка модифікованного метода Дамерау-Левенштейна.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) слайди презентації

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз методив обробки природної	1.04.2022	
	мови		
2.	Аналіз методів визначеня семантичної	15.04.2022	
	близкості текстів		
3.	Розробка модифікованного метода	30.04.2022	
	Дамерау-Левенштейна		
4.	Оформлення матеріалів кваліфікаційної роботи	4.05.2022	
5.	Подання кваліфікаційної роботи керівникові та	8.05.2022	
	її попередній захист		
6.	Подання роботи на рецензування	14.05.2022	

Дата видачі завдання 28 березня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Фесенко Т.Г..
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 79 с., 33 рис., 2 дод., 20 джерел.

КОЕФІЦІЄНТ БЛИЗКОСТІ ТЕКСТУ, NLP, TELEGRAM СЕРВІС, ТОКЕНІЗАЦІЯ, СТЕМІНГ, МЕТОД ДАМЕРАУ-ЛЕВЕНШТЕЙНА.

Метою кваліфікаційної роботи є розробка Telegram бота для перевірки автентичності інформації, отриманої з новинних телеграм каналів, використовуючи методи визначення текстової близькості.

Мають бути вирішені наступні задачі:

- реалізувати взаємодії системи з офіційними, достовірними джерелами;
- проаналізувати отриману інформацію;
- надати користувачу відсоток достовірності вхідної інформації.

ABSTRACT

Master’s thesis: 79 pages, 33 figures, 2 appendices, 20 sources.

TEXT PROXIMITY RATIO, NLP, TELEGRAM SERVICE, TOKENIZATION, STAMMING, DAMERAU-LEVENSHTEIN METHOD.

The major goal of this thesis is to develop a Telegram bot to verify the authenticity of information obtained from news telegrams of channels, using methods of determining text proximity.

The following tasks must be solved:

- implement system interactions with official, reliable sources;
- analyze the information received;
- provide the user with a percentage of the reliability of the input information.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

NLP (Natural Language Processing) - загальний напрямок штучного інтелекту та математичної лінгвістики. Він вивчає проблеми комп'ютерного аналізу та синтезу текстів природними мовами.

RESR API (Representational State Transfer Application Programming Interface) - архітектурний стиль взаємодії компонентів розподіленої програми у мережі.

БД (База даних) - сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування

IDE (Integrated development environment) - комплекс програмних засобів, що використовується програмістами для розробки програмного забезпечення.

LDA (Latent Dirichlet allocation) - застосовувана в машинному навчанні та інформаційному пошуку породжувальна модель, що дозволяє пояснювати результати спостережень за допомогою неявних груп, завдяки чому можливе виявлення причин схожості деяких частин даних.

SVD (Singular Value Decomposition) - факторізацією дійсної або комплексної матриці.

ВСТУП

Метою роботи є систематизувати відомі досягнення в галузі семантичного аналізу. Вивчення способів комплексного застосування вихідних та розроблених методів у галузі семантичного аналізу інформації.

Семантика - розділ лінгвістики, що вивчає смислове значення одиниць мови. Крім знань про структуру мови, семантика тісно пов'язана з філософією, психологією та іншими науками, оскільки неминуче торкається питань про походження значень слів, їх відношення до буття та мислення. При семантичному аналізі необхідно враховувати соціальні та культурні особливості носія мови. Процес людського мислення, як і мова, яка є інструментом вираження думок, є дуже гнучким і важко піддається формалізації. Тому семантичний аналіз вважається найскладнішим етапом автоматичної обробки текстів.

Створення нових методів семантичного аналізу текстів відкриє нові можливості та дозволить істотно просунутися у вирішенні багатьох завдань комп'ютерної лінгвістики, таких як машинний переклад, автореферування, класифікація текстів та інших. Не менш актуальною є розробка нових інструментів, що дозволяють автоматизувати семантичний аналіз.

На даний момент існує кілька методів уявлення сенсу висловлювань, однак жоден із них не є універсальним. Над співвіднесенням сенсу тексту працювало багато дослідників. Так, І.А. Мельчук запровадив поняття лексичної функції, розвинув поняття синтаксичних та семантичних валентностей та розглянув їх у контексті тлумачно-комбінаторного словника, який є мовною моделлю. Він показав, що значення слів співвідносяться не безпосередньо з навколишньою дійсністю, а з уявленнями носія мови про цю дійсність.

Більшість дослідників схиляються до думки, що семантичний аналіз повинен виконуватися після синтаксичного. Найважливішими є обов'язкові

рольові зв'язки, далі йдуть зв'язки кореференції, потім факультативні рольові зв'язки і потім предметно-асоціативні.

Універсальна мова представлення знань має бути зручним інструментом для виведення нових знань з наявних, а значить, необхідно створити апарат для перевірки правильності висловлювань. Ймовірно, що у напрямі створення подібних семантичних мов розвиватиметься наукова думка у майбутньому.

Незважаючи на те, що деякі наукові та технічні ідеї в галузі обробки текстів розвиваються досить швидко, багато проблем семантичного аналізу залишаються невирішеними. Більшість дослідників дійшли висновку, що словник для підтримки семантичного аналізу має оперувати смислами і, отже, описувати властивості та відносини понять, а не слів. Але постає питання, як правильно структурувати і подавати інформацію в подібних словниках, щоб пошук за ними був зручним і швидким, до того ж була б можливість враховувати зміни в природній мові (зникнення старих та виникнення нових понять).

В даний час спостерігається велике зростання обсягів інформації, що зберігається та обробляється в базах даних (БД) та інформаційних системах, мережах Інтернет та Інтранет. При цьому досить важливим і поширеним є завдання виявлення та усунення дубльованої та суперечливої інформації. В основі її вирішення лежить теорія інтелектуального аналізу даних. При вирішенні зазначеної задачі та інших завдань інтелектуальної обробки розглядаються різні варіанти прикладних областей та різні підходи.

1 ОГЛЯД ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Обґрунтування актуальності обраної теми

Найпоширенішими джерелами отримання інформації є Telegram канали. За останні кілька років він набрав популярності та навіть вийшов на один рівень з Facebook. Але в наш час більшість соціальних мереж публікують неправдиву інформацію та поширюють пропаганду. Telegram - канали новин із десятками тисяч користувачів подають користувачам неправдиву інформацію, дублюють сітку мовлення державних каналів, емоційно коментують висловлювання політиків. Люди навіть не замислюються про перевірку достовірності даних і, в результаті, отримана дезінформація впливає на поведінку та дії людини, її емоційний стан [1].

Саме тому запропонована робота направлена на розробку Telegram бота який, в свою чергу, буде допомагати користувачам Telegram каналів фільтрувати отриману інформацію. Робота буде будуватися з використанням методів розділення та обробки текстових даних. Також будуть розглядатися різні методи та покращуватися вже існуючі за допомогою аналізу різних наборів даних.

1.2 Огляд існуючих аналогів

1.2.1 Система контролю версій

Одними з найпоширеніших систем порівняння тексту є системи порівняння програмного коду на наявність змін або системи контролю версій. Системи контролю версій – це програмні інструменти, які допомагають командам розробників керувати змінами у вихідному коді. У світлі ускладнення середовищ розробки вони допомагають командам

розробників працювати швидше та ефективніше [2].

Програмне забезпечення контролю версій відстежує всі зміни, що вносяться в код, у спеціальній базі даних. Порівняння тексту зазвичай виконується посимвольно. Тобто, якщо зміниться хоч оди символ у рядку, система помітить зміни та виділить цей символ.

При виявленні помилки розробники можуть повернутися назад і порівняти з попередніми версіями коду для виправлення помилок, зводячи до мінімуму проблеми для всіх учасників команди. Код проекту, програми або програмного компонента зазвичай організований у вигляді структури папок або дерева файлів. Один розробник у команді може працювати над новою можливістю, а інший у цей час змінювати код для виправлення незв'язаної помилки, тобто кожен розробник може вносити свої зміни в кілька частин дерева файлів [3].

Контроль версій допомагає командам вирішувати подібні проблеми шляхом відстеження кожної зміни, внесеної кожним учасником, та запобігати виникненню конфліктів при паралельній роботі.

1.2.2 Системи перекладу та розпізнавання тексту

Для спрощення роботи з текстом розробники програмного забезпечення створили спеціальні програми, що дозволяють автоматизувати введення великих обсягів текстових даних. Також текст великими обсягами можна не лише вводити, а й перекладати. Для автоматизації процесів роботи з текстом використовуються системи перекладу та розпізнавання тексту [4].

Вводити інформацію в комп'ютер можна не лише з клавіатури, але й за допомогою спеціального пристрою – сканера. У процесі сканування текст із журналу або книги з паперового формату переводиться в електронний. Спочатку відсканований текст має вигляд графічного зображення, тобто сприймається комп'ютером як зображення. Для того, щоб з картинки отримати текстовий формат і далі працювати з нею, як з текстом,

використовуються спеціальні програми, що виконують розпізнавання тексту.

Процес розпізнавання відбувається так [5]:

1) програма аналізує отримане зображення, виділяючи у ньому текстові, табличні та графічні області;

2) потім рядки в текстових блоках розбиваються на окремі слова, слова – розбиваються на символи;

3) потім кожен символ порівнюється з зображенням букв, цифр або спеціальних символів, що є в базі;

4) знайшовши оптимальний варіант, програма видає його користувачеві як розпізнаний текст.

Високий рівень розвитку технологій, що забезпечують реалізацію інформаційних процесів зберігання та пошуку інформації, сприяв популяризації програм-перекладачів. Програма-перекладач є програмним продуктом, який дозволяє здійснювати переклад з однієї мови на іншу окремих слів, словосполучень та речень [4]. Дія таких систем перекладу будується на розбитті вхідного тексту на окремі токени, потім застосовуються правила побудови словосполучень та речень природної мови. Перекладач аналізує текст початковою мовою, а потім складає такий же текст новою мовою.

1.2.3 Огляд існуючих систем

Git - абсолютний лідер за популярністю серед сучасних систем управління версіями. Це розвинений проект з активною підтримкою та відкритим вихідним кодом.

Система використовується безліччю професійних розробників програмного забезпечення. Вона чудово працює під управлінням різних операційних систем і може застосовуватися з безліччю вбудованих середовищ розробки (IDE) [6].

Git показує дуже високу продуктивність у порівнянні з великою

кількістю альтернатив. Це можливо завдяки оптимізації процедур фіксації коммітів, створення гілок, злиття та порівняння попередніх версій. Алгоритми Git розроблені з урахуванням глибокого знання атрибутів, притаманних реальним деревам файлів вихідного коду, і навіть типової динаміки змін та послідовностей доступу.

Деякі системи керування версіями керуються іменами файлів під час роботи з деревом файлів та ведення історії версій. Замість обробки назв система Git аналізує вміст. Це важливо, оскільки файли вихідного коду часто перейменовують, розділяють та змінюють місцями. Об'єктні файли репозиторію Git формуються за допомогою дельта-кодування (фіксації відмінностей вмісту) та компресії. Крім того, такі файли в чистому вигляді зберігають об'єкти з вмістом каталогу та метаданими версіями [7]. Приклад роботи Git наведений на рисунку 1.1.

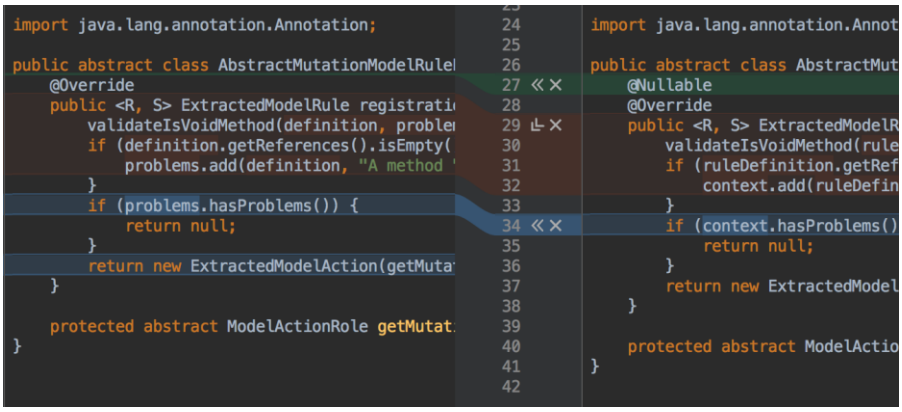
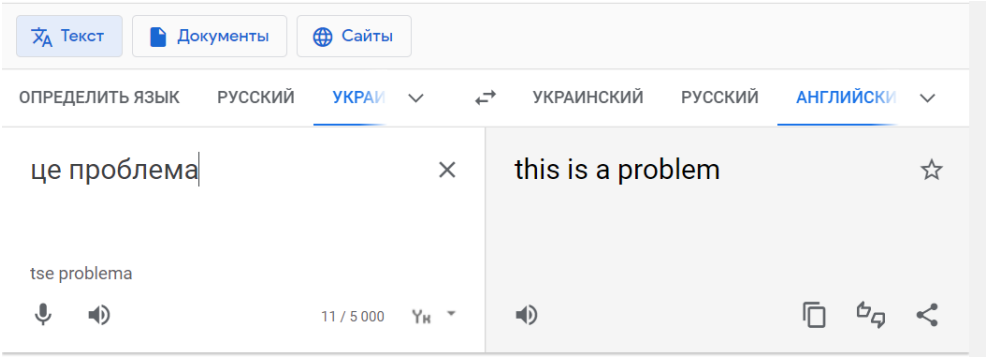


Рисунок 1.1 – Приклад роботи Git

Перекладач Google - це сервіс для перекладу, і при цьому це майже інструмент для спілкування з іноземцями, автономний перекладач і водночас путівник. В його основі лежить переклад промовлених фраз, розробники вже створюють систему перекладу «на льоту», в ньому є офлайн-словники для роботи без доступу до Мережі та переклад дорожніх знаків, на яких достатньо навести камеру свого мобільного пристрою.

Google Translate лише в автономному режимі пропонує понад 80 мов, що підключаються через налаштування. Він став одним з найбільш доступних перекладачів, що практично вгадують побудову речення при перекладі, значення слів і при цьому сервіс може навчатися [8]. Приклад роботи Google Translate наведений на рисунку 1.2.



Рисуноку 1.2 – Приклад роботи Google Translate

Таблиця 1.1 – Порівняння систем аналізу тексту

Системи	**Переваги**	**Недоліки**
Система управління версіями Git	Запам'ятовування попереднього стану аналізованого тексту	Можливі конфлікти під час аналізу тексту
	Впровадження нового тексту не зачіпаючи загальну структуру файлу	Посимвольний аналіз змін у файлі, що у певних випадках може некоректно обробляти зміни
Перекладач Google	Виділення та розбиття тексту на речення, токени	Є обмеження на кількість символів
	Можливість роботи з документами, картинками, голосом	Некоректно визначає мову введеного тексту

Перекладач, до того ж, володіє приємним інтерфейсом, озвученням перекладених слів, фраз і цілих текстів. Крім цього, при доступі до Мережі сервіс пропонує розпізнавання тексту, аудіо, рукописне введення. Він здатний вибудовувати цілі діалоги, а при перекладі веб-сторінок не порушує їх форматування.

1.3 Мета та постановка задачі

Метою роботи є розробка Telegram бота для перевірки автентичності інформації, отриманої з новинних телеграм каналів, використовуючи методи визначення текстової близькості.

Для досягнення поставленої мети, мають бути вирішені наступні задачі:

- реалізувати взаємодії системи з офіційними, достовірними джерелами;
- проаналізувати отриману інформацію;
- надати користувачу відсоток достовірності вхідної інформації.

За допомогою REST API (Representational State Transfer Application Programming Interface) організовано взаємодію компонентів системи для надання інформації користувачеві (рисунок 1). На стороні сервера виконується попередня обробка тексту підготовленої новини та подальший аналіз на основі методів визначення текстової близькості. У системі передбачено використання бази даних, з якою взаємодіє сервер та зберігання результату аналізу вихідного тексту для подальшої публікації новини у новинному каналі.

2 АНАЛІЗ ІСНУЮЧИХ ТЕХНОЛОГІЙ З ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Огляд NLP методів

Обробка природної мови (NLP) – це розділ науки про дані, який займається текстовими даними. Крім числових даних, значною мірою доступні текстові дані, що використовуються для аналізу та вирішення бізнес-завдань. Але, перш ніж використовувати дані для аналізу або прогнозування, важливо обробити дані. Щоб підготувати текстові дані для побудови моделі над текстом проводиться попередня обробка. На рисунку 2.1 наведено основні методи для обробки текстових даних.

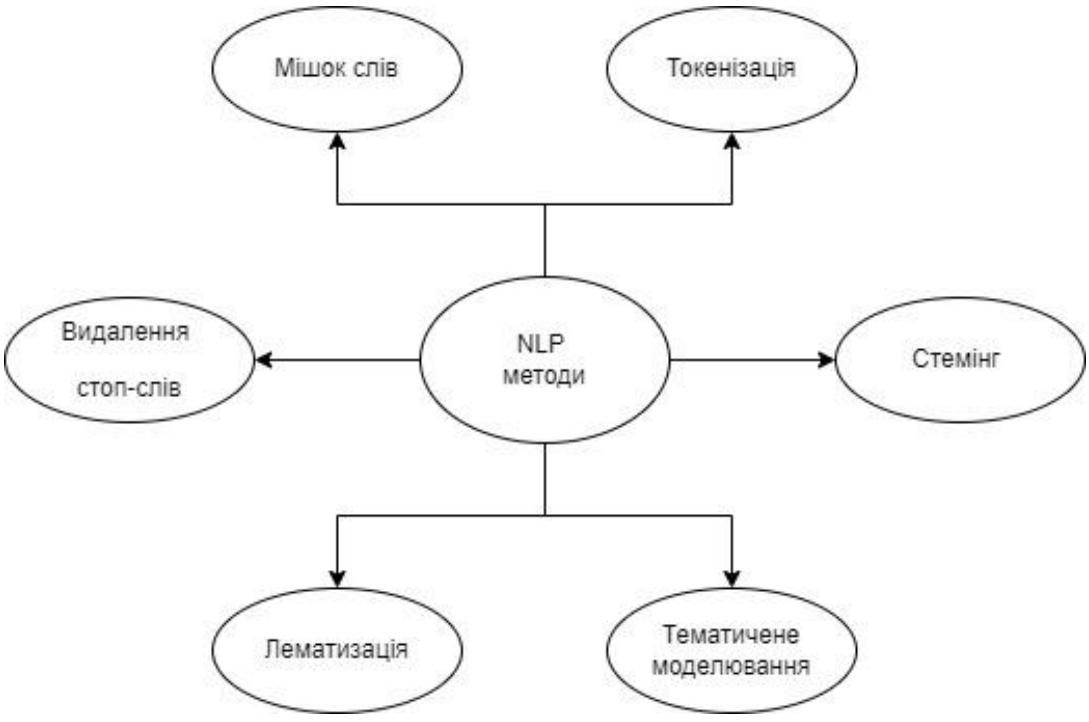


Рисунок 2.1 – NLP методи

2.1.1 Мішок слів

Це часто використовувана модель, яка дозволяє підраховувати всі слова у фрагменті тексту. По суті, він створює матрицю входжень для речень або документа, ігноруючи граматику і порядок слів. Ці частоти слів що зустрічаються, потім використовуються в якості ознак для навчання класифікатора.

Цей підхід може відображати кілька недоліків, таких як відсутність семантичного значення і контексту, а також факти, які зупиняють слова (наприклад, «the» або «a»), додають ускладнення в аналіз, а деякі слова не мають відповідної ваги («всесвіт» важить менше, ніж слово «вони»).

Щоб вирішити цю проблему, один з підходів полягає в тому, щоб змінити частоту слів в залежності від того, як часто вони з'являються в усіх текстах (а не тільки в тому, який ми аналізуємо), щоб оцінити вживання слів, таких як « the », що також часто зустрічаються в інших текстах. Цей підхід до оцінки називається «Частота терміну - зворотна частота документа» (TFIDF), і він покращує набір слів за вагою. Через TFIDF терміни, що часто зустрічаються в тексті «винагороджуються» (наприклад, слово «вони» в нашому прикладі), але вони також і «караються», якщо ці терміни часто зустрічаються в інших текстах, які ми також включаємо в алгоритм. Навпаки, цей метод виділяє і «нагороджує» унікальні терміни з урахуванням всіх текстів. Проте, у цього підходу немає ні контексту, ні семантики.

2.1.2 Токенізація

Це процес розбиття тексту на речення і слова. По суті, це завдання розрізати текст на частини, названі токенами, і в той же час відкинути певні символи, такі як знаки пунктуації [9]. Приклад токенізації зображено на рисунку 2.2.

Рисунок 2.2 – Приклад токенізації

Токенізація також може видалити знаки пунктуації, полегшуючи шлях до правильної сегментації слів, але також викликаючи можливі ускладнення. У разі точок, наступних за абревіатурою (наприклад, dr.), період, наступний за цим скороченням, слід розглядати як частину одного і того ж знаку і не видаляти.

Процес токенізації може бути особливо проблематичним при роботі з біомедичними текстовими доменами, які містять безліч дефісів, дужок чи інших знаків пунктуації.

2.1.3 Видалення стоп-слів

Метод включає в себе позбавлення від загальномовних артиклів, займенників і прийменників, таких як «і», «the» або «to» англійською мовою. У цьому процесі деякі дуже поширені слова, які, мабуть, не мають великого значення для мети НЛП або не мають ніякого значення, фільтруються і виключаються з оброблюваного тексту, тим самим видаляючи широко поширені і ті терміни, що часто трапляються, які не інформативні для відповідного тексту [10] .

Стоп-слова можна безпечно ігнорувати, виконуючи пошук в заздалегідь заданому списку ключових слів, звільняючи місце в базі даних і скорочуючи час обробки.

Універсального списку стоп-слів не існує. Їх можна вибрати заздалегідь або створити з нуля. Потенційний підхід - почати з прийняття заздалегідь визначених стоп-слів і додати слова до списку пізніше. Проте, схоже, що

останнім часом загальна тенденція полягала в тому, щоб перейти від використання великих стандартних списків стоп-слів до використання взагалі без списків.

Справа в тому, що видалення стоп-слів може стерти релевантну інформацію і змінити контекст в даному реченні. Наприклад, якщо ми виконуємо аналіз настроїв, ми можемо збити наш алгоритм зі шляху, якщо видалимо стоп-слово, наприклад «не». У цих умовах ви можете вибрати мінімальний список стоп-слів і додати додаткові терміни в залежності від вашої конкретної мети.

2.1.4 Стемінг

Відноситься до процесу розрізання кінця або початку слова з метою видалення афіксів (лексичних доповнень до кореня слова). Проблема в тому, що афікси можуть створювати або розширювати нові форми одного і того ж слова (так звані флективні афікси) або навіть самі створювати нові слова (так звані словотворчі афікси). В англійській мові префікси завжди є похідними (афікс створює нове слово, як в прикладі з префіксом «eco» в слові «ecosystem»), але суфікси можуть бути похідними (афікс створює нове слово, як в прикладі з суфікс «ist» в слові « guitarist ») або флективними (афікс створює нову форму слова, як в прикладі з суфіксом« er »в слові« faster »). Приклад роботи стемінгу зображено на рисунку 2.3.

Рисунок 2.3 — Приклад стемінгу

Можливий підхід полягає в тому, щоб розглянути список загальних афіксів і правил (мови Python і R мають різні бібліотеки, що містять афікси і методи) і виконати підставу на них, але, звичайно, цей підхід має обмеження. Оскільки Стеммер використовують алгоритмічні підходи, результатом процесу стемінг може бути не справжнє слово, ні навіть зміна значення слова (і речення). Щоб компенсувати цей ефект, ви можете редагувати ці зумовлені методи, додаючи або видаляючи афікси і правила, але ви повинні враховувати, що ви можете покращувати продуктивність в одній області, виробляючи погіршення в інший. Завжди дивіться на картину цілком і перевіряйте працездатність своєї моделі.

Стеммер прості у використанні і працюють дуже швидко (вони виконують прості операції з рядком), і якщо швидкість і продуктивність важливі в моделі NLP, то стемінг, безумовно, кращий варіант [11].

2.1.5 Лематизація

Має на меті привести слово до його базової формі і згрупувати різні форми одного і того ж слова. Наприклад, дієслова в минулому часі замінюються справжнім (наприклад, «пішов» замінений на «йти»), а синоніми уніфіковані (наприклад, «кращий» замінений на «хороший»), тим самим стандартизуючи слова з аналогічним значенням їх кореня. Хоча це здається тісно пов'язаним з процесом утворення коренів, лематизація використовує інший підхід для досягнення кореневих форм слів.

Наприклад, слова «running», «runs» і «ran» є формами слова «run», тому «run» - це лема всіх попередніх слів. Приклад роботи лематизації зображено на рисунку 2.4.

Рисунок 2.4 – Приклад лематизації

Лематизація також бере до уваги контекст слова для вирішення інших проблем, таких як усунення неоднозначності, що означає, що вона може розрізняти ідентичні слова, які мають різні значення в залежності від конкретного контексту. Лематизація - завдання набагато більш ресурсномістке, ніж виконання процесу стемінг. У той же час, оскільки для цього потрібно більше знань про структуру мови, ніж для підходу з виділенням коренів, він вимагає більшої обчислювальної потужності, ніж настройка або адаптація алгоритму виділення залишків.

2.1.6 Тематичне моделювання

Є методом виявлення прихованих структур в наборах текстів чи документів. По суті, він групує тексти, щоб виявляти приховані теми на основі їх змісту, обробляючи окремі слова і привласнюючи їм значення на основі їх розподілу. Цей метод заснований на припущенні, що кожен документ складається з суміші тем і що кожна тема складається з набору слів, а це означає, що, якщо ми можемо виявити ці приховані теми, ми можемо розкрити зміст наших текстів.

З всесвіту методів тематичного моделювання, ймовірно, найбільш часто використовується прихований розподіл Діріхле (LDA). Цей відносно новий алгоритм (винайдений менше 20 років назад) працює як метод навчання без вчителя, розкриває різні теми, що лежать в основі набору документів. В таких методах навчання без учителя, як цей, немає вихідної

змінної, яка б спрямовувала процес навчання, і дані досліджуються алгоритмами для пошуку закономірностей. Щоб бути більш конкретним, LDA знаходить групи пов'язаних слів по:

- Призначенню кожного слова випадковій темі, де користувач визначає кількість тем, які він хоче розкрити. Ви не визначаєте самі теми (ви визначаєте тільки кількість тем), і алгоритм буде зіставляти всі документи з темами таким чином, щоб слова в кожному документі в основному охоплювалися цими уявними темами.

- Алгоритм перебирає кожне слово ітеративно і перепризначає слово темі, беручи до уваги ймовірність того, що слово належить темі, і ймовірність того, що документ буде створений темою. Ці ймовірності обчислюються багаторазово, до збіжності алгоритму.

На відміну від інших алгоритмів кластеризації, таких як K-засоби, які виконують жорстку кластеризацію (де теми не перетинаються), LDA призначає кожному документу суміш тем, це означає, що кожен документ може бути описаний однією або декількома темами (наприклад, Документ 1 описується 70% теми A, 20% теми B і 10% теми C) і відображає більш реалістичні результати. Приклад тематичного моделювання зображено на рисунку 2.5.

Рисунок 2.5 – Приклад тематичного моделювання

Тематичне моделювання надзвичайно корисно для класифікації текстів, побудови рекомендаційних систем (наприклад, щоб рекомендувати вам

книги на основі ваших минулих читань) або навіть для виявлення тенденцій в онлайн-публікаціях.

2.2 Детальний огляд методу токенізації

Токенізація — один з перших кроків в NLP, і це завдання розбиття послідовності тексту на блоки з семантичним значенням. Ці одиниці називаються токенами, і складність токенізації полягає в тому, як отримати ідеальний поділ, щоб всі токени в тексті мали правильне значення і не було пропущених токенів.

У більшості мов текст складається зі слів, розділених пробілами, де окремі слова мають семантичне значення. Приклад розбиття тексту зображено на рисунку 2.6.

Рисунок 2.6 – Приклад розбиття тексту

«burger» — це вид їжі, «and» — союз, «good» — позитивне прикметник і т.д. Таким чином, токенізація кожного елемента має значення, і, об'єднавши всі значення кожного токену, ми можемо зрозуміти значення всього речення. Знаки пунктуації також отримують свої власні маркери, кому для поділу речення і крапку, щоб позначити кінець речення.

На рисунку 2.7 представлений альтернативний вид токенізації.

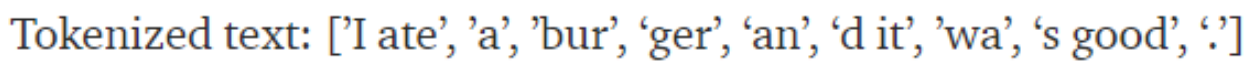


Рисунок 2.7 – Альтернативний вид розбиття тексту

Для багатослівної одиниці «I ate» ми можемо просто додати значення «I» і «ate», а підслівні одиниці «bur» і «ger», не мають значення окремо, але, об'єднавши їх, ми приходимо до знайомого слова, і ми можемо зрозуміти, що воно означає.

Також ми можемо зробити висновок, що «it» — займенник, а буква «d» належить попередньому слову. Але після цієї токенізації попереднє слово «an» вже має значення в англійській мові, артикль «an» сильно відрізняється від «and». Таким чином можна дотримуватися слів і додати знакам пунктуації їх власні символи. Це найбільш поширений спосіб токенізації, званий токенізацією на рівні слів [12].

2.2.1 Токенізація на рівні слів

Цей метод складається тільки з поділу речення на пробіли і знаки пунктуації. В Python є безліч бібліотек, які роблять це, включаючи NLTK, SpaCy, Keras, Gensim, або ви можете створити власне Regex.

Поділ на пробіли також може розділити елементи, які слід розглядати як один токен, наприклад, «New York». Це проблематично, і в основному це відбувається з іменами, запозиченими іноземними фразами і сполуками, які іноді записуються як кілька слів [13].

Ще один недолік токенізації на рівні слів - це величезний словниковий запас, який вона створює. Кожен токен зберігається в словниковому запасі токенів, і якщо словник складається з усіх унікальних слів, що містяться в усьому вхідному тексті, він створює величезний словниковий запас, що в подальшому призводить до проблем з пам'яттю і продуктивністю. Сучасна

архітектура глибокого навчання Transformer XL має словниковий запас 267735. Щоб вирішити проблему великого словникового запасу, можна подумати про створення токенів з символами замість слів, що називається токенізацією рівня символу.

2.2.2 Токенізація на рівні символів

Вперше представлений Karpathy в 2015 році, замість поділу тексту на слова, поділ виконується на символи. Розмір словникового запасу різко скорочується до кількості символів на мові, 26 для англійської мови плюс спеціальні символи. Орфографічні помилки або унікальні слова обробляються краще, тому що вони розбиті на символи, і ці символи вже відомі в словнику.

Зменшення розміру словника залежить від довжини послідовності. Тепер, коли кожне слово розбивається на всі його символи, токенізована послідовність набагато довше вихідного тексту. Слово «smarter» перетворено в 7 різних токенів. Крім того, не досягається основна мета токенізації, оскільки символи, принаймні, в англійській мові, не мають семантичного значення. Тільки при об'єднанні символів вони знаходять сенс. В якості проміжного блоку між токенізацією слів та токенізацією символів створюється проміжний блок підслова, менший ніж слово, але більший ніж просто символ.

2.2.3 Токенізація на рівні підслова

Токенізація на рівні підслова перетворює найбільш поширені слова і розбиває унікальні слова на значущі підслівні одиниці. Приклад роботи зображено на рисунку 2.8. Якби «unfriendly» був позначений як унікальне слово, воно було б розкладено на «un-friend-ly», де всі є значущими одиницями, «un» означає протилежне, «friend» - іменник, а «ly» перетворює

його в прислівник. Проблема тут в тому, як зробити цю сегментацію, як розбити на «un-friend-ly», а не «unfr-ien-dly».

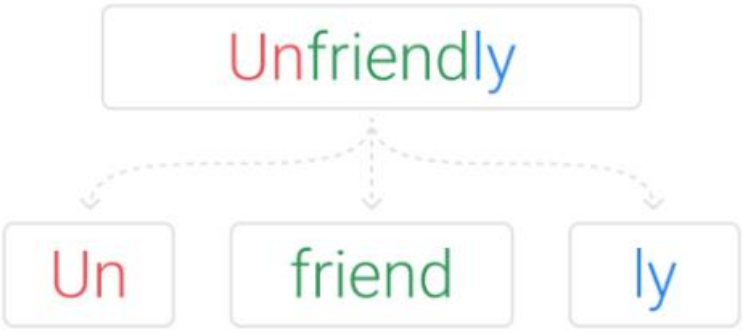


Рисунок 2.8 — Приклад токенізації підслова

Починаючи з 2020 року, сучасні архітектури глибокого навчання, засновані на Transformers, використовують токенізацію на рівні підслова. Модель мовного уявлення BERT виконує наступну токенізацію зображену на рисунку 2.9.

Raw text: I have a new GPU.

Tokenized text: ['i', 'have', 'a', 'new', 'gp', '##u', '.']

Рисунок 2.9 – Приклад токенізації BERT

Слова, присутні в словнику, токенізуються саме як слова, але «GPU» не зустрічається в словнику і розглядається як унікальне слово. Дотримуючись алгоритму, вирішено, що він сегментований на «gp-u». Знак ## перед 'u' означає, що це підслово належить до того ж слова, що і попереднє підслово. BPE, Unigram LM, WordPiece і SentencePiece є найбільш поширеними алгоритмами токенізації.

2.3 Визначення семантичної близькості термів

Семантична близькість термів вже давно є невід'ємною частиною теорії обробки текстів природною мовою [14]. Семантична близькість між двома сутностями з часом може змінюватися у зв'язку зі змінами словників. Крім того, семантична близькість двох сутностей може бути різною в різних предметних областях.

Семантична близькість термів може бути обчислена за допомогою спеціалізованих баз даних та статистичних корпусів. Також велика кількість сучасних підходів до обчислення семантичної близькості ґрунтується на обчисленні відстаней між словами.

Методами визначення семантичної близькості текстів можна виділити:

- метод порівняння текстів;
- метод латентно-семантичного аналізу;
- метод визначення подібності;
- метод Дамерау-Левенштейна.

2.3.1 Метод порівняння текстів

Вирішення завдань повнотекстового пошуку області інформаційно-аналітичної обробки текстів виходить із зіставлення текстів та оцінки їх подібності шляхом порівняння різних ознак. Спочатку основним способом порівняння текстів було їх зіставлення за ключовими словами.

Класичні алгоритми інформаційного текстового пошуку розглядають запит користувача, як безліч ключових слів і впорядковують документи, що містять слова запиту, за спаданням інформаційної значущості цих слів у знайдених текстах [15]. Однак для пошукових запитів, сформульованих у вигляді речень або фраз, а також при запитально-відповідному та термінологічному пошуку, векторна модель далеко не завжди дає якісні результати. Причина цього полягає в тому, що методи на основі векторної

моделі не враховують інформацію про зв’язки та відносини слів у тексті. При цьому якісне вирішення завдань пошуково-аналітичної обробки, таких, як автоматичне анотування текстів, виявлення текстових запозичень, питання-відповідь пошук, неможливо без урахування лінгвістичної інформації.

Сучасні методи інформаційного пошуку та порівняння текстів враховують такі критерії при зіставленні текстів один з одним:

- спільну зустрічальність, порядок слів запиту та відстань між словами в текстах документів;
- синтаксичні та семантичні зв'язки між словами запиту;
- предметну область та тематику текстів;
- відносини між словами тексту в тезаурусі або онтології предметної галузі (синонімічні, родовидові та ін.).

Критерії першої групи відносяться до нелінгвістичних. Вони не вимагають проведення лінгвістичного аналізу (за винятком морфологічного, який застосовується нині практично у всіх системах інформаційного пошуку для нормалізації слів та дозволу омонімії) [16].

Критерії другої групи ґрунтуються на результатах синтаксичного та семантичного аналізу для встановлення відповідних характеристик текстів. Облік лінгвістичної інформації дозволяє реалізувати питання-відповідь в режимі пошуку і сприяє підвищенню точності порівняння текстів.

Критерії третьої групи припускають наявність таксономії (безліч тематичних рубрик), заданої апріорно або побудованої автоматично, наприклад, шляхом кластеризації. Порівняння текстів проводиться з урахуванням інформації про їхню тематичну приналежність.

Критерії четвертої групи вимагають застосування лінгвістичних онтологій для встановлення зв'язків між словами та термінами (концептами) предметних областей. При порівнянні текстів враховуються пов'язані терміни, як і у зіставлюваних текстах.

Таким чином, основною тенденцією в галузі методів пошуку та аналізу текстової інформації є облік безлічі різних факторів з метою надання

користувачеві пошуково-аналітичних сервісів, що мають високу точність і повноту.

Якщо порівнювати роботу та ефективність методу на різній кількості даних та з різними вхідними значеннями можна виділити результати, наведені у таблиці 2.1.

Таблиця 2.1 – Результати методу порівняння текстів

Кількість слів у реченні	17	44	60
15	0.777	0.311	0.229
40	0.39	0.68	0.515
53	0.372	0.665	0.707

Отже, з отриманих даних можна зробити висновок, що при, приблизно, однаковій кількості вхідних слів коефіцієнт близькості текстів збільшується. На рисунку 2.10 зображено графік залежності коефіцієнтів від вхідних даних.

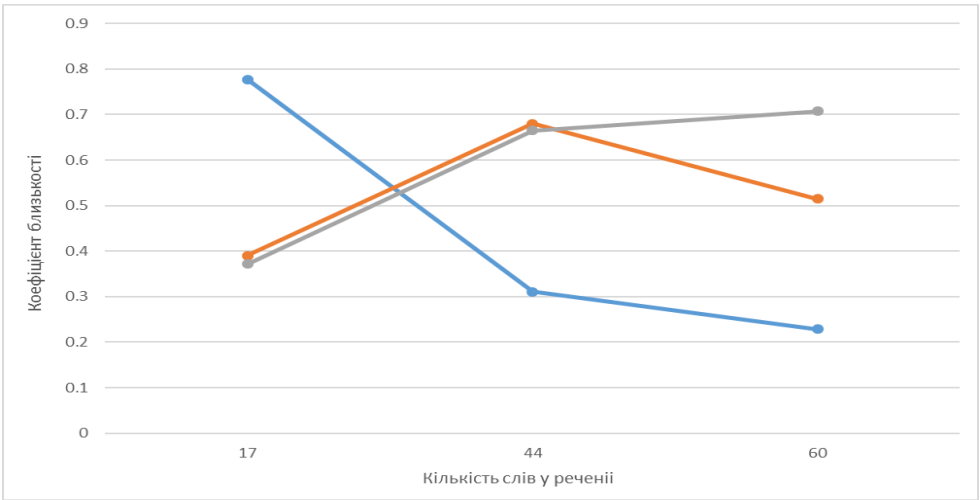


Рисунок 2.10 – Графік залежності вхідних даних та коефіцієнтів близькості при аналізі методу порівняння тексту

2.3.2 Метод латентно-семантичного аналізу

В якості вихідної інформації ЛСА використовує матрицю терми-на-документи (терми – слова, словосполучення; документи – тексти, класифіковані або за будь-яким критерієм, або розділені довільним чином – це залежить від задачі), що описує набір даних, що використовується для навчання системи. Елементи цієї матриці містять, як правило, ваги, що враховують частоту використання кожного терма в кожному документі або ймовірні заходи (PLSA – імовірнісний латентно-семантичний аналіз), що базуються на незалежному мультимодальному розподілі [17].

Найбільш поширений варіант ЛСА заснований на використанні розкладання речовиннозначної матриці за сингулярним значенням або SVD-розкладання (SVD – Singular Value Decomposition). За допомогою нього будь-яку матрицю можна розкласти у безлічі ортогональних матриць, лінійна комбінація яких є досить точним наближенням до вихідної матриці.

Основна ідея латентно-семантичного аналізу полягає у наступному:

після перемноження матриць отримана матриця, що містить тільки k перших лінійно незалежних компонент вихідної матриці A, що відображає структуру залежностей (в даному випадку асоціативних), латентно присутніх у вихідній матриці. Структура залежностей визначається ваговими функціями термів кожного документа.

Вибір k залежить від поставленого завдання та підбирається емпірично. Він залежить від кількості вихідних документів.

Якщо документів небагато, наприклад, сотня, то k можна брати 5-10% від загальної кількості діагональних значень; якщо документів сотні тисяч, то беруть 0,1-2%. Слід пам'ятати, що якщо обране значення k дуже велике, то метод втрачає свою потужність і наближається за характеристиками до стандартних векторних методів. А надто маленьке значення k не дозволяє помітити різницю між схожими термами чи документами: залишитися лише одна головна компонента, яка «перетягне на себе одіяло», тобто. все слабко і

навіть незв'язані терми.

Застосування зосереджено на:

- порівнянні двох термів між собою;
- порівнянні двох документів між собою;
- порівнянні терму та документу.

Іноді цей метод використовують для знаходження «найближчого сусіда» — найбільш близьких за вагою термів, асоціативно пов'язаних із вихідним. Цю властивість використовують для пошуку близьких за змістом термів.

Перевагою методу вважається його чудова здатність виявляти залежності між словами, коли звичайні статистичні методи безсилі. ЛСА також може бути застосований, як з навчанням (з попередньою тематичною класифікацією документів), так і без навчання (довільне розбиття довільного тексту), що залежить від завдання, що вирішується.

Для прикладу можна застосувати порівняння двох термів. Вхідні речення поділяються на терми та видаляються терми, які не інформативні для відповідного тексту, знаходиться коефіцієнт близькості текстів. Отриманні дані після аналізу алгоритму наведені у таблиці 2.2.

Таблиця 2.2 – Результат метод латентно-семантичного аналізу

Кількість слів у реченні	17	44	60
15	0.839	0.451	0.285
40	0.423	0.735	0.685
53	0.392	0.712	0.852

В результаті аналізу отриманих результатів можна зробити висновок, що при поділі вхідних речень на терми та видаленні зайвих частин, коефіцієнти подібності тексту зросли. Також відповідний графік наведено на рисунку 2.11.

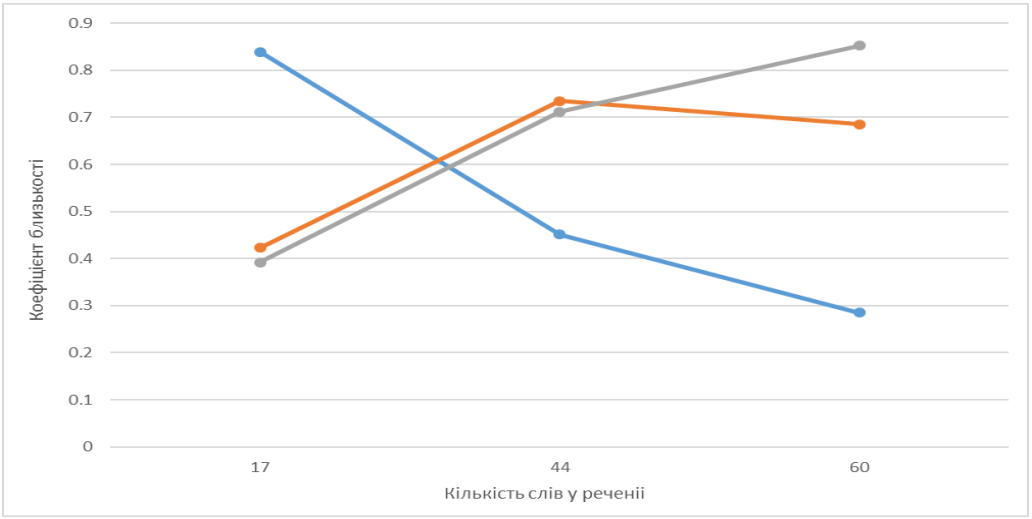


Рисунок 2.11 – Графік залежності вхідних даних та коефіцієнтів близькості при використанні методу латентно-семантичного аналізу

2.3.3 Метод визначення подібності

Для статистичних даних, таких як частота вживання слів, схожість слів або речень, використовуються методи більшою мірою незалежні від мови. До таких методів належить алгоритм швидкого пошуку подібності.

Оцінка семантичної подібності текстів (Semantic Text Similarity–STS) складається з набору попарних речень, які пов'язують оцінками від 0 до 5, для визначення їхньої семантичної пов'язаності, де 0 означає, що пропозиції не пов'язані, а 5 вказує на ідентичність речень на семантичному рівні [18].

Працюючи з великими масивами, що містять мільйони різних словоформ, необхідний високоефективний алгоритм обчислення загальних рис, оскільки слова необхідно порівнювати по-парним способом. Для цього завдання було обрано алгоритм швидкого пошуку подоби (Fast Similarity Search algorithm (FastSS)).

Вимірювання семантичного подібності текстів є темою обробки природних мов, де використовується велика різноманітність варіантів. Знаходження семантично схожого змісту текстів призведе до розширення випадків використання базового елемента пошуку для отримання інформації.

Однак, крім вилучення інформації, багато інших додатків, які мають справу з текстовими даними, можуть отримати вигоду з алгоритмів текстової подібності та асоційованих індексних структур. Наприклад, у поштовій програмі, це може використовуватися, щоб показати електронні листи, які стосуються одного користувача, що він читає чи пише у даний час. Цей метод також може бути використаний, як основа для текстової кластеризації, автоматичної фільтрації, автоматичної класифікації повідомлень електронної пошти в папки, і навіть передбачення одержувача або верифікації.

Одним з можливих методів для скорочення кількості порівнянь є скорочений зворотній індекс, який пов'язує термін з обмеженим набором джерел. Але у будь-якого виду скорочення є ризик втрати важливої інформації через звичайні перепустки. Це може призвести до скорочення кількості вже знайдених подібних елементів. За відповідних сприятливих умов, з погляду семантичної подоби, якість скорочень знайдених пунктів може навіть збільшиться. Але стоїть проблема у комп'ютеризації представлення відповідностей. Часто це зроблено за допомогою заходів статистичного значення. Інший аспект виміру семантичної схожості полягає в тому, що, навіть без будь-якого скорочення, є ризик втрати документів тієї ж теми, в яких використовуються інші схожі за змістом слова.

Системи аналізу текстової подоби складаються з багатьох компонентів. Головна ідея, що лежить в основі даної моделі, полягає в тому, щоб традиційні модулі обробки природної мови (такі як розпізнавання тимчасових та іменних сутностей) вдавалися до неї тільки після ретельної попередньої обробки. За допомогою цієї моделі можна знайти типи текстів і блоків, і використовувати аналітичні алгоритми тільки тоді, коли вони мають сенс. Цей підхід знижує кількість помилок до рівня порівнянного з тими, що описані в літературі, а також зберігає рівень помилок відносно постійним.

Для аналізу роботи алгоритму вхідний текст розбивався на окремі блоки, та дані вираховувалися отриманими блоками. Таким чином в випадках, коли кількість вхідних даних була майже однакова, коефіцієнт

подібності збільшувався. Результати алгоритму наведені у таблиці 2.3.

Таблиця 2.3 – Результати методу визначення подібності

Кількість слів у реченні	17	44	60
15	0.812	0.325	0.211
40	0.325	0.863	0.421
53	0.231	0.431	0.893

У графічному виді дані з таблиці наведені на рисунку 2.12.

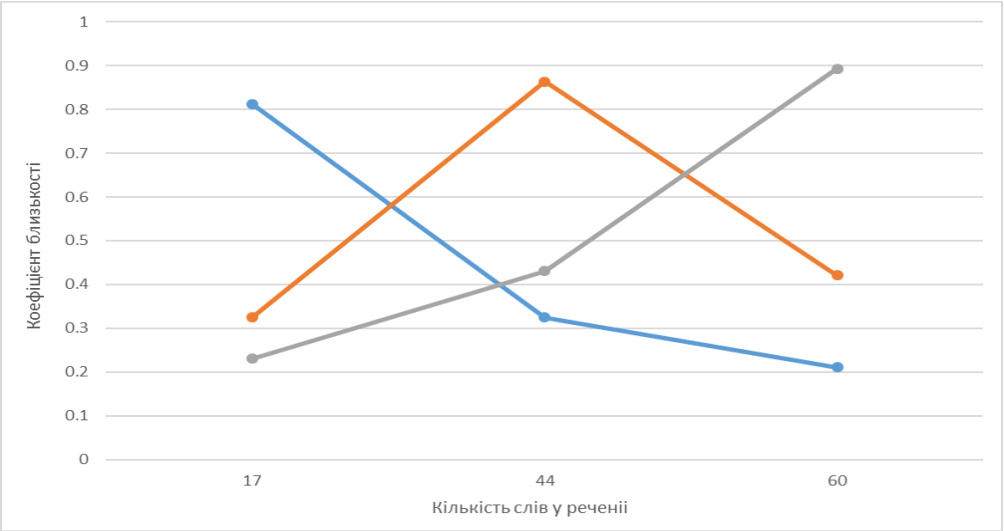


Рисунок 2.12 – Графік залежності вхідних даних та коефіцієнтів близькості при використанні методу визначення подібності

2.3.4 Метод Дамерау-Левенштейна

З'ясування того, наскільки схожі два рядки, а потім перетворення цієї подібності на кількісне вимірювання, є основною проблемою в аналізі тексту та обробці природної мови. Існує декілька ефективних методів вирішення

цієї проблеми. Один з них - відстань Левенштейна, як міра відстані редагування, простий, але широко використовуваний метод [19].

Відстань редагування між двома рядками — це мінімальна загальна кількість операцій, які перетворюють один рядок на інший. Відстань редагування дорівнює нулю, якщо два рядки ідентичні. Відстань редагування є позитивною, якщо два рядки не ідентичні. Відстань редагування не враховує довжину порівнюваних рядків: два рядки, близькі по довжині і відрізняються одним символом, обробляються так само, як два рядки, один з яких набагато довший за інший.

Для аналізу роботи алгоритму вхідний текст аналізуємо, виділяємо максимально схожі речення та слова, які є різними, перетворюємо за допомогою метода Дамерау-Левенштейна. Таким чином в випадках, коли кількість слів у вхідному реченні була менша або приблизно дорівнювала кількості слів у вхідному тексті, коефіцієнт подібності збільшувався. Результати алгоритму наведені у таблиці 2.4.

Таблиця 2.4 – Результати методу Дамерау-Левенштейна

Кількість слів у реченні	17	44	60
15	0.892	0.835	0.864
40	0.324	0.863	0.847
53	0.231	0.331	0.901

У графічному виді дані з таблиці наведені на рисунку 2.13.

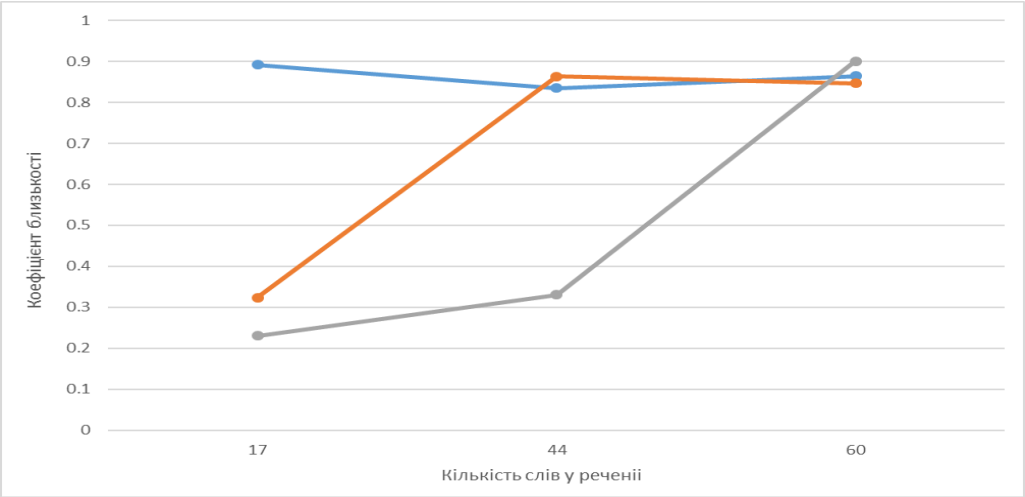


Рисунок 2.13 – Графік залежності вхідних даних та коефіцієнтів близькості при використанні методу Дамерау-Левенштейна

2.4 Аналіз обраного метода

2.4.1 Відстань Левенштейна

Відстань Левенштейна обчислює відстань редагування за допомогою операцій вставки, видалення та заміни. Це найекономічніший набір операцій для перетворення одного рядка на інший. Відстань Левенштейна можна обчислити рекурсивно, але у ефективних реалізаціях використовуються рішення динамічного програмування з таблицею зберігання обчислених витрат.

На рисунку 2.14 показано операції редагування відстані. Перші чотири приклади демонструють операції з одним символом та порожніми рядками:

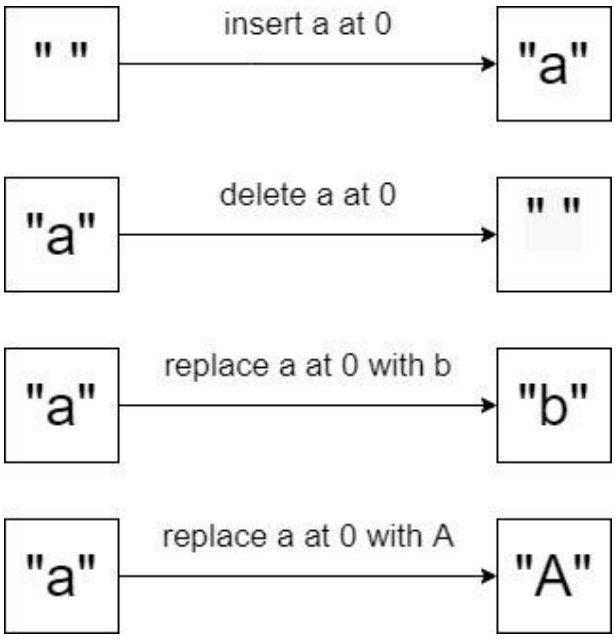


Рисунок 2.14 - Операцій редагування відстані

Потім у нас є проста послідовність операцій заміни, зображених на рисунку 2.15, які перетворюють «ab» в «ba» :

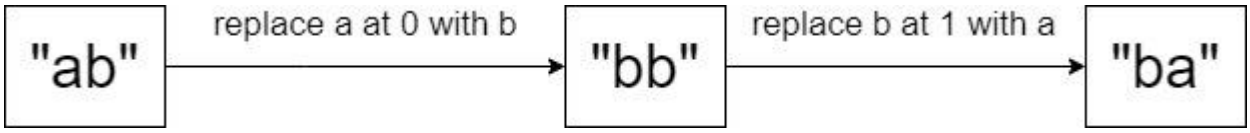


Рисунок 2.15 – Операція заміни

В обох цих прикладах відстань редагування дорівнює 3, тому що для перетворення одного слова в інше потрібно три операції: На рисунку 2.16 зображено приклад знаходження відстані.

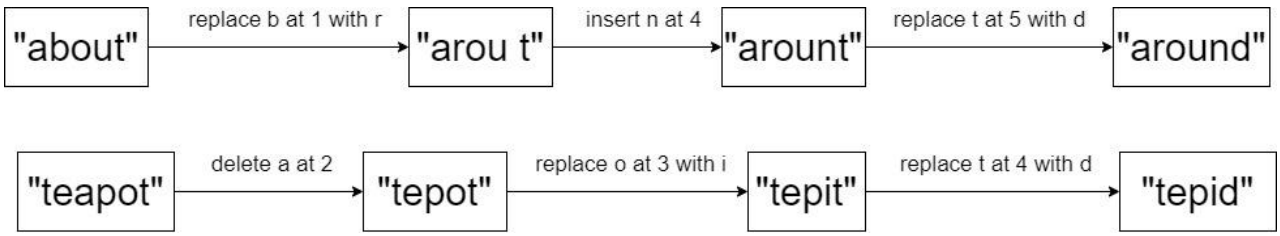


Рисунок 2.16 – Приклад знаходження відстані

2.4.2 Відстань Дамерау-Левенштейн

Дамерау-Левенштейн розширює спосіб відстані Левенштейна додатковою операцією, а саме: транспонувати, тобто, де можна поміняти місцями два сусідні знаки. Метод може обчислити менші відстані редагування, якщо суміжні операції транспонування можуть замінити кілька операцій заміни. Ефективні реалізації також ґрунтуються на динамічному програмуванні. На рисунку 2.17 зображено послідовність операцій заміни за допомогою методу Дамерау-Левенштейна [20].

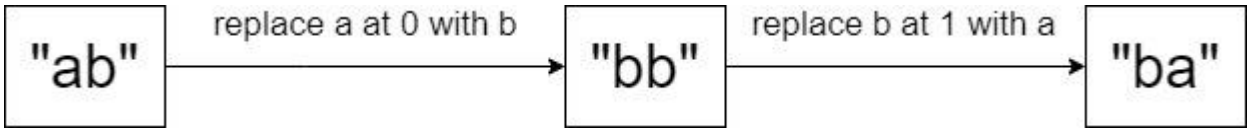


Рисунок 2.17 – Операції заміни за допомогою методу Дамерау-Левенштейна

Ще раз поглянемо на подібність «ab» і «ba», де відстань редагування Левенштейна дорівнює 2: На рисунку 2.18 зображено операцію транспонування.

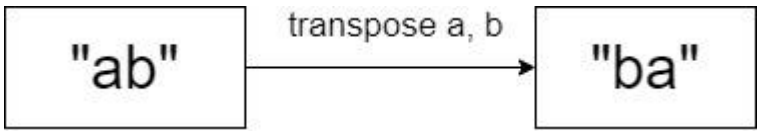


Рисунок 2.18 – Операція транспонування символів

Відстань редагування між «роздратуванням» і «ризиком» більше з відстанню Левенштейна, ніж Дамерау-Левенштейна. Інший спосіб подумати про це: "роздратування" і "ризик" більш схожі при використанні Дамерау-Левенштейна, ніж при використанні Левенштейна. На рисунку 2.19 зображено приклади роботи алгоритмів.

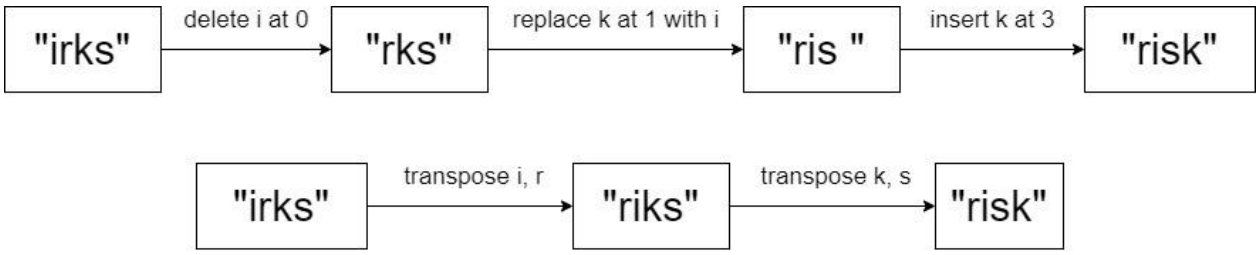


Рисунок 2.19 – Приклади перетворення слів

2.4.3 Розбір та реалізація алгоритму

Розрахунки Дамерау-Левенштейна виконуються шляхом розрахунку відстані редагування, показаного на рисунку 2.20, з неправильним словом MALAGN і цільовим словом MALANG. Оцінка значення відстані редагування виходить із перетину кожного рядка та стовпця. Обчислення починається з позиції індексу першого та останнього рядка та стовпця. Результати можуть бути відомі після розрахунку до кінця стовпця, що буде значенням відстані редагування. Нижче наведено значення відстані редагування, яка обчислюється в кінці рядка та стовпця.

S/T		M	A	L	A	N	G
	0	1	2	3	4	5	6
M	1	0	1	2	3	4	5
A	2	1	0	1	2	3	4
L	3	2	1	0	1	2	3
A	4	3	2	1	0	1	2
G	5	4	3	2	1	1	1
N	6	5	4	3	2	1	1

Рисунок 2.20 – Таблиця відстані редагування

Процес в алгоритмі Дамерау Левенштейна виглядає так:

1) ініціалізація n для довжини символу s та m для довжини символу t. Якщо $n = 0$ або $m = 0$, то повернути значення у вигляді відстані редагування у формулу: $edit\_rate = max\ (n, \text{м})$, потім перейдіть до кроку 7;

2) створення матриці d з $m + 1$ рядка та $n + 1$ стовпця;

3) перший рядок містить $0..n$ та заповнює перший стовпець $0..m$;

4) перевірка кожного символу від s до t. Якщо $s[i] = t[j]$, то вартість $= 0$. Якщо $s[i] \neq t[j]$, то вартість $= 1$;

5) заповніть значення кожного осередку $d\ [i,j]$ рядково: $d[i,j] = min(x,y,z)$ пояснення: $d[i,j]$: комірка, яка показує стовпець, що зустрічається з рядком і в матриці d.

x : значення, яке знаходиться у верхній комірці від поточної позиції комірки плюс 1 (один) або може бути сформульовано: $x = d[i - 1,j] + 1$.

y : значення в комірці ліворуч від поточної позиції осередку плюс 1 (один) або може бути сформульовано так: $y = d[i,j - 1] + 1$.

z : значення, що міститься у верхній комірці лівої комірки тепер (північний захід) плюс значення вартості і може бути сформульовано: $z = d[i - 1,j - 1] +$ вартість.

Якщо $i > 1$ і $j > 1$ і $s[i] = t[j - 1]$ і $s[i - 1] = t[j]$. Це означає, що після порівняння двох слів є символи, які можна транспонувати, потім заповніть значеннях комірок $d\ [i,j]$ за такою формулою: $d[i,j] = min(d[i,j], d[i - 2,j - 2] +$ вартість).

6) далі слід визначити розраховану відстань редагування, яку можна

знайти у комірках $d[n, m]$ у правому кутку останнього рядка;

7) закінчення алгоритму.

На рисунку 2.21 зображено блок-схему алгоритму роботи програми з використанням методу Дамерау Левенштейна.

Рисунок 2.21 – Блок-схема алгоритму роботи програми

2.5 Обраний алгоритм для роботи

Порівняння методів визначення семантичної близькості текстів наведено на рисунку 2.22. Критеріями порівняння методів є коефіцієнт близькості тексту залежний від кількості слів у реченні.

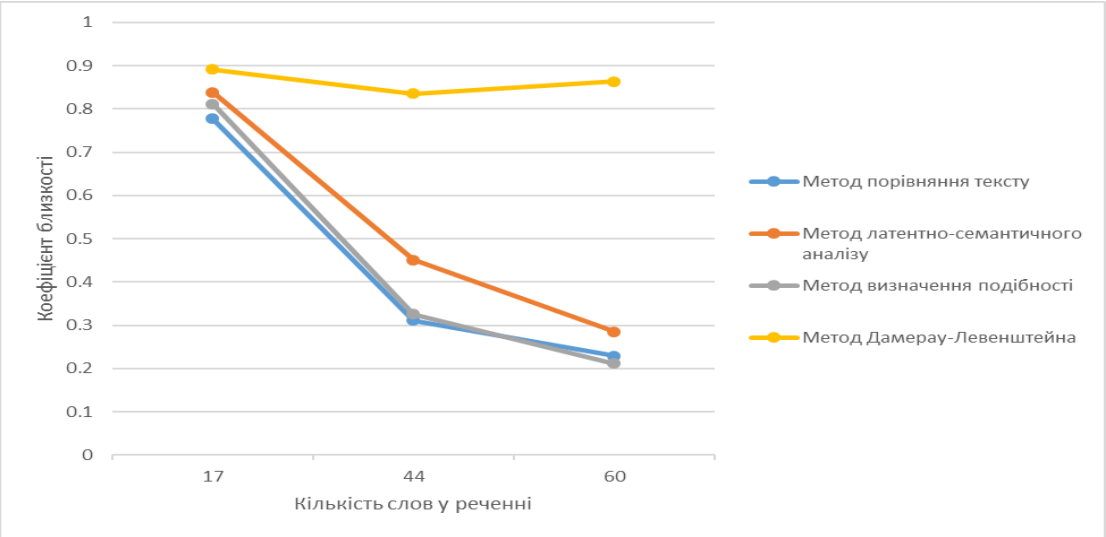


Рисунок 2.22 – Порівняння методів

Виходячи з отриманих результатів для перетворення тексту в роботі було обрано метод Дамерау Левенштейна. В роботі використовується модифікований метод за допомогою прибирання функцій додавання та перестановки символів.

Аналізуємий тест розбивається по реченням, далі аналізується вже окреме речення. За допомогою методу токенезації речення буде розбите на окремі слова і в окремому слові за допомогою методу стемінгу тексту у слові будуть видалені префікси та закінчення. Отримане слово буде порівнюватися зі схожим словом за допомогою модифікованого метода Дамерау Левенштейна.

Завдяки видаленню приставки та закінчення у слові метод Дамерау Левенштейна працює ефективніше та швидше, адже не використовуються методи вставки та перестановки символів. Також таке рішення зменшить відстань між словами та більш чітко визначить однакові слова.

3 АНАЛІЗ СИСТЕМИ ТА ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1 Запропонована система

В роботі запропонована система з використанням сервісу Telegram. Схема наведена на рисунку 3.1.

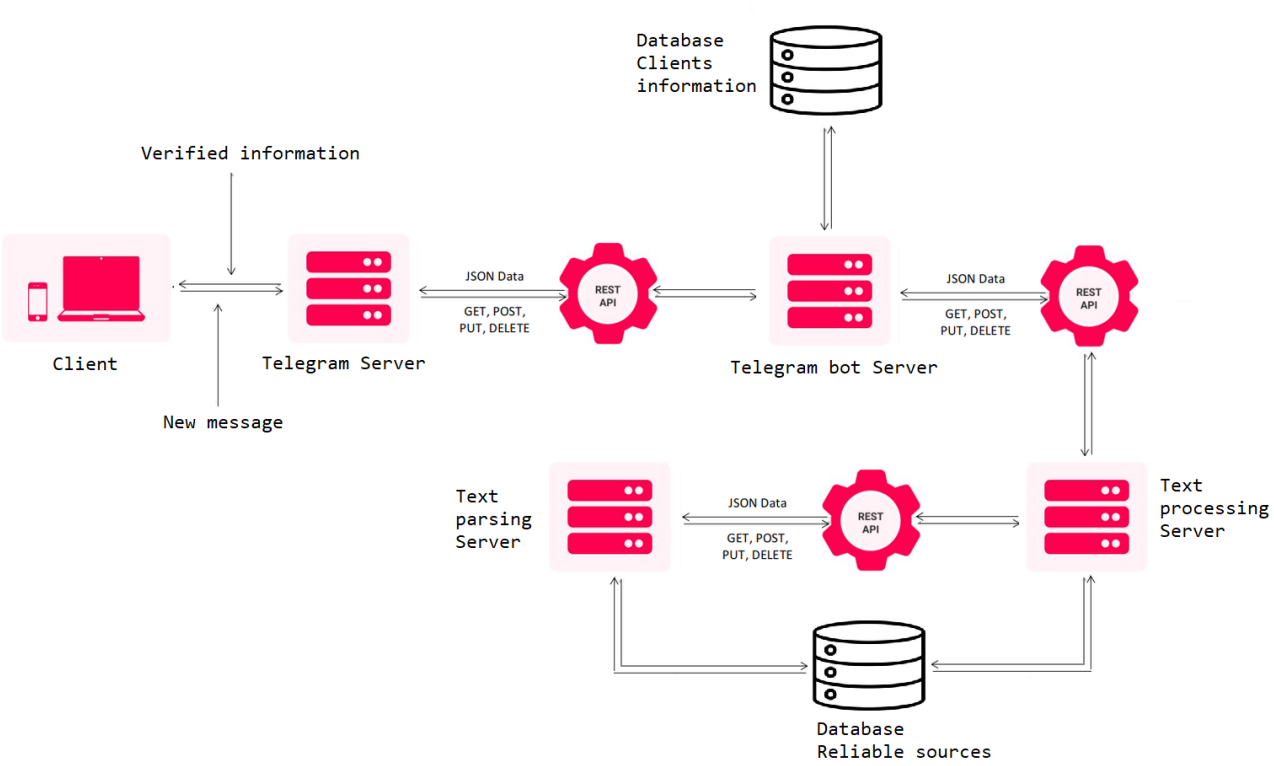


Рисунок 3.1 – Схема розробленої системи

Но вхід дана система отримує нове повідомлення від користувача та передає його до Telegram серверу. Telegram це кросплатформова система миттєвого обміну повідомленнями (месенджер), що дозволяє обмінюватися текстовими, голосовими та відеоповідомленнями, стікерами та фотографіями, файлами багатьох форматів.

Отримане повідомлення за допомогою REST API відправляється на Telegram bot сервер. На стороні сервера є Client information база даних у якій

зберігається інформація про користувачів Telegram бота. Інформація в базі верифікується та взаємодіє з Telegram ботом за допомогою SQL запитів.

Після отримання та верифікації користувача отримане повідомлення за допомогою REST API відправляється на Text processing сервер. На стороні сервера повідомлення обробляється з використанням NLP методів. За еталонне значення береться інформація з Reliable sources бази даних у якій зберігаються новини, отримані з офіційних джерел інформації. Також з базою взаємодіє Text parsing сервер, яких парсить інформацію з офіційних джерел інформації та за допомогою SQL запитів доповнює базу.

3.2 Метод Дамерау-Левенштейна Text processing сервера

У Text processing сервері для обробки та порівняння окремого слова використовується модифікований метод Дамерау-Левенштейна.

Основними операціями прямого метода Дамерау-Левенштейна є:

- вставка;
- заміна;
- видалення;
- перестановка.

Для модифікації такий метод обраний через те, що більш чітко визначає однакові слова в тексті за допомогою визначення значення відстані між словами. Також метод може виключити можливість орфографічних помилок. Таким чином в системі знижується людський фактор помилок в тексті.

Для модифікації метода було використано метод стемінгу слова. Завдяки цьому кожне слово обробляється функцією, яка видаляє закінчення та префікси у слові. Таким чином в методі Дамерау-Левенштейна є можливість не використовувати операції вставки та перестановки символів. Це впливає на значення відстані між словами, що дає можливість чітко визначати однакові слова. Отже для обробки слова кращім є цей метод для

виключення орфографічних помилок.

Але дана модифікація впливає на час обробки слова. Метод стемінгу уповільнює роботу сервера, через додаткові операції над словом та подвійному аналіз кожного слова. Ця проблема вирішується розпаралелюванням на графічних процесорах операцій обробки слів.

3.3 Блок схема алгоритму

Обробка та аналіз вхідних значень реалізовані у Text processing сервері. Отримане вхідне повідомлення розбивається на окремі речення та виділяються максимально схожі речення з еталонним текстом. Для цього реалізований метод sameSentences() наведений у лістингу 3.1.

Лістинг 3.1 – Реалізація методу sameSentences()

```
        public static String sameSentences(String sentence,
List<String> list) {
            List<String> sentenceList =
Arrays.stream(sentence.split(" "))
                                                .toList();
            String returnSentences = null;
            int numberOfTheSameWords = 0;

            for (String s : list) {
                List<String> sList = Arrays.stream(s.split(" "))
                                            .toList();

                int count = 0;

                for (String i : sentenceList) {
                    for (String j : sList) {
                        if (i.equals(j)) {
                            count++;

                            break;
                        }
                    }
                }

                if (numberOfTheSameWords < count) {
                    numberOfTheSameWords = count;
```

```
                    returnSentences = s;
                }
            }

            return returnSentences;
        }
```

Однакові речення за допомогою методу токенізації розбиваються на окремі слова (токени). Речення, як масив слів, обробляється за допомогою метода стемінгу. Реалізація методу stemming(), для видалення префіксу та закінчення у слові наведена у лістингу 3.2.

Лістинг 3.2 – Реалізація методу stemming()

```
        private static List<String> stemming(List<String>
inputList) {
            List<String> outputList = new ArrayList<>();

            String[] endingList = {"ов", "ев", "єв", "им", "ом",
"ов", "ин", "ін", "у", "ю", "ові", "еві", "єві", "і", "ї"};

            String[] prefixesList = {"з", "с", "без", "від",
"од", "між", "над", "об", "під", "перед", "понад", "пред", "роз"
                    , "через", "пре", "при", "прі"};

            for (String word : inputList) {
                String outputValue = word;

                for (String end : endingList){
                    if(word.endsWith(end)){
                        outputValue = outputValue.replace(end,
"");

                        break;
                    }
                }

                for(String prefix : prefixesList){
                    if(word.startsWith(prefix)){
                        outputValue                          =
outputValue.replace(prefix, "");

                        break;
                    }
                }
```

```
                outputList.add(outputValue);
            }

            return outputList;
        }
```

Для порівняння та знаходження однакових слів використовується модифікований метод Дамерау-Левенштейна. Реалізація методу DamerauLevenshteinDistance() наведена у лістингу 3.3.

Лістинг 3.3 – Реалізація методу DamerauLevenshteinDistance()

```
     public static int DamerauLevenshteinDistance(String str1,
String str2){
    int[] Di_1 = new int[str2.length() + 1];
    int[] Di = new int[str2.length() + 1];

    for (int j = 0; j <= str2.length(); j++) {
        Di[j] = j; // (i == 0)
    }

    for (int i = 1; i <= str1.length(); i++) {
        System.arraycopy(Di, 0, Di_1, 0, Di_1.length);

        Di[0] = i; // (j == 0)

        for (int j = 1; j <= str2.length(); j++) {
            int cost = (str1.charAt(i - 1)!= str2.charAt(j -
1))?1 : 0;

            Di[j] = min(
                    Di_1[j] + 1,
                    Di[j - 1] + 1,
                    Di_1[j - 1] + cost
            );
        }
    }

    return Di[Di.length - 1];
}
```

Реалізація методу textProcessing() з використанням методів stemming() та DamerauLevenshteinDistance() для обробки слів наведена у лістингу 3.4.

Лістинг 3.4 – Реалізація методу textProcessing()

```
         public static OptionalDouble textProcessing(String
inputText, String referenceText) {
             List<String> inputList =
Arrays.stream(inputText.split("\\."))
                                                .toList();
             List<String> referenceList =
Arrays.stream(referenceText.split("\\."))
                                                    .toList();
             List<Double> proximityFactorList = new
ArrayList<Double>();

             for (String inputSentence : inputList) {
                 String sameSentence =
sameSentences(inputSentence, referenceList);

                 List<String> inputWords =
Arrays.stream(inputSentence.split(" "))
                                                     .toList();
                 List<String> sameSentenceWords =
Arrays.stream(sameSentence.split(" "))

.toList();

                 List<String> processInputWords =
stemming(inputWords);

                 List<String> processSameSentenceWords =
stemming(sameSentenceWords);

                 int sameWordCount = 0;

                 for (String word1 : processInputWords) {
                     for (String word2 :
processSameSentenceWords) {
                         int distance =
DamerauLevenshteinDistance(word1, word2);

                         if (distance < 3) {
                             sameWordCount++;

                             break;
                         }

                     }
                 }


proximityFactorList.add(proximityFactor(inputWords.size(),
sameSentenceWords.size(), sameWordCount));
             }
```

```
        return proximityFactorList.stream()
                                  .toList()
                                  .stream()
                                  .mapToDouble(e -> e)
                                  .average();
    }
```

Завдяки використанню метода stemming() збільшується відсоток близькості вхідного тексту та зменшується час використання методу DamerauLevenshteinDistance() через те, що порівнюються вже оброблені слова. Таким чином методи перестановки та видалення зайвих символів у слові не використовуються.

Алгоритм роботи системи:

1) початок;
2) на вхід система отримує нове повідомлення від користувача як еталонне значення;
3) вхідний текст розбивається на окремі речення за допомогою методу токенізації;
4) порівнюються вхідні тексту та виділяються подібні речення;
5) подібні речення розділяються на окремі слова за допомогою методу токенізації;
6) для видалення закінчення та префіксу слова речення обробляється за допомогою методу стемінга;
7) окремі слова обробляються за допомогою модифікованого методу Дамерау-Левенштейна та визначаються однакові слова;
8) вихідне значення: коефіцієнт близкості вхідних текстів;
9) кінець

Блок схема алгоритму наведена на рисунку 3.2.

Рисунок 3.2 – Блок схема роботи Text processing сервера

4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Аналіз результатів

Для аналізу будемо оцінювати результати розробленої системи з використанням методу Дамерау-Левенштейна та модифікованого методу Дамерау-Левенштейна з використанням семінгу слова. Критеріями для аналізу є: відсоток близкості вхідних даних та час, витрачений на обробки вхідних даних.

Тест 1: валідні вхідні значення.

Для вхідних значень будемо використовувати новину з офіційного сайту та опубліковану новину у Telegam каналі. В таблиці 4.1 наведені вхідні дані для тестування системи.

Таблиця 4.1 – Вхідні значення

Новина з офіційного сайту (еталонне значення)	Президент США Джо Байден 9 травня має підписати закон про ленд-ліз для України. Аналогічна програма працювала вісімдесят років тому, саме під час Другої світової війни.
Новина у каналі (вхідний текст)	Джо Байден підписав закон про постачання Україні зброї за ленд-лізом, повідомили у Білому домі.

На рисунку 4.1 зображено новину з Telegam-каналу.

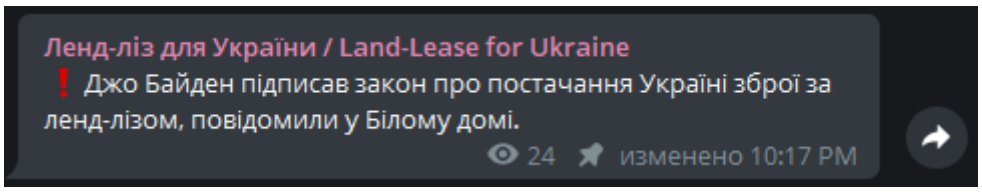


Рисунок 4.1 – Новина з Telegam-каналу

В результаті використання методу Дамерау-Левенштейна система видає результат близькості тексту у 73.9%. Тобто можна враховувати, що новина не є пропагандою. Результати зображені на рисунку 4.2.

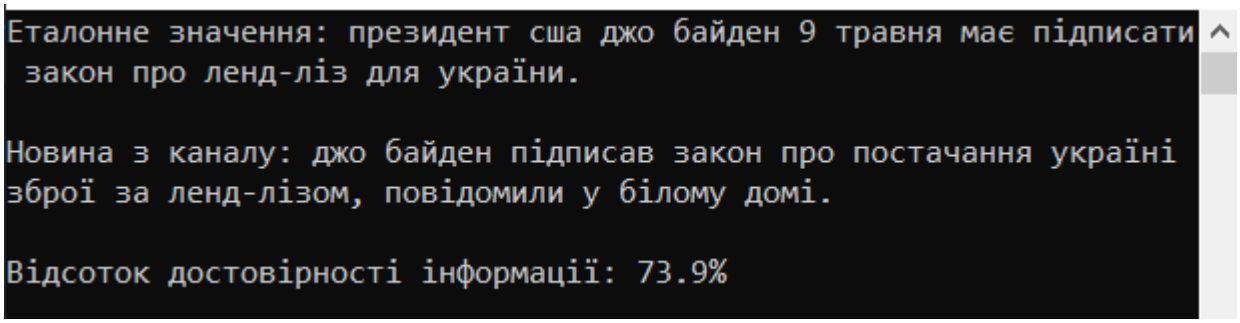


Рисунок 4.2 – Результат системи з використанням методу Дамерау-Левенштейна

Далі розглядається система з використанням модифікованого методу Дамерау-Левенштейна. Для аналізу використовуються значення зазначені у таблиці 4.1.

В результаті використання модифікованого методу Дамерау-Левенштейна система видає результат близькості тексту у 89.5%. Таким чином модифікований метод видає більш чіткі значення близькості тексту. Результати зображені на рисунку 4.3.

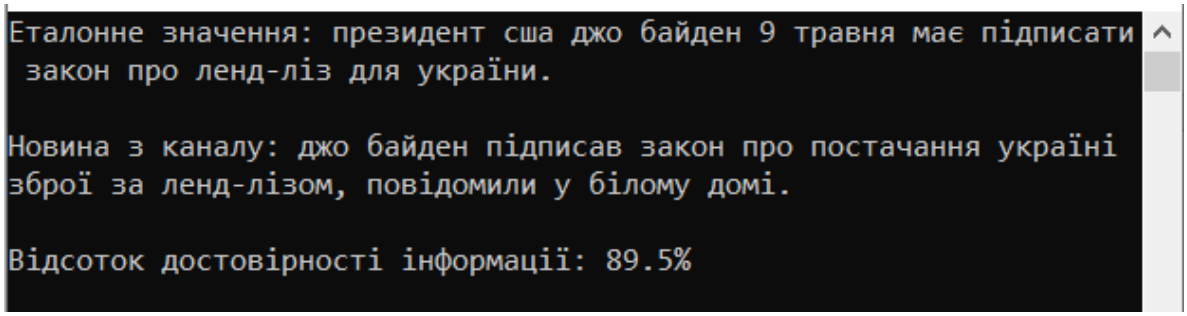


Рисунок 4.3 – Результат системи з використанням модифікованого методу Дамерау-Левенштейна

Аналіз часу витрачений на алгоритм наведений у таблиці 4.2.

Таблиця 4.2 – Час витрачений на обробку даних

Метод	Час
Метод Дамерау-Левенштейна	766 мкс.
Модифікований метод Дамерау-Левенштейна	814 мкс

Тест 2: не валідні вхідні значення.

Також проаналізована система на невалідних вхідних значеннях. Вхідні дані наведені у таблиці 4.3.

Таблиця 4.3 – Невалідне повідомлення

Невалідне значення (вхідний текст)	Джо Байден відмовився підписувати закон про ленд-ліз.

Еталонним значенням для цього повідомлення можна вважати значення з таблиці 4.1.

В результаті використання методу Дамерау-Левенштейна система видає результат близькості тексту у 35.4%. Тобто можна враховувати, що

новина є неправдивою. Результати зображені на рисунку 4.4.

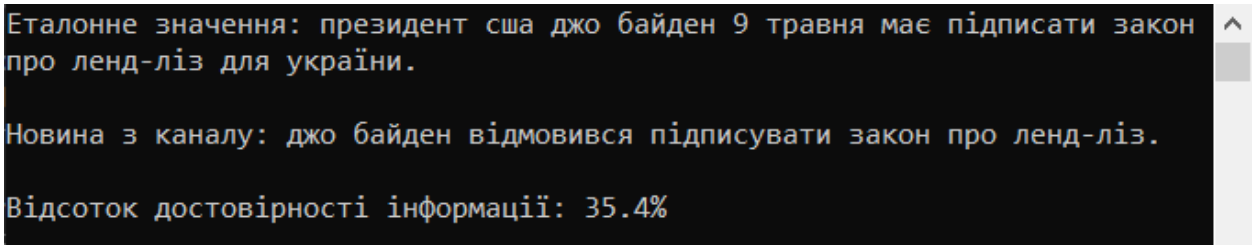


Рисунок 4.4 – Результат системи з використанням методу Дамерау-Левенштейна

В результаті використання модифікованого методу Дамерау-Левенштейна система видає результат близькості тексту у 40.6%. Результати зображені на рисунку 4.5.

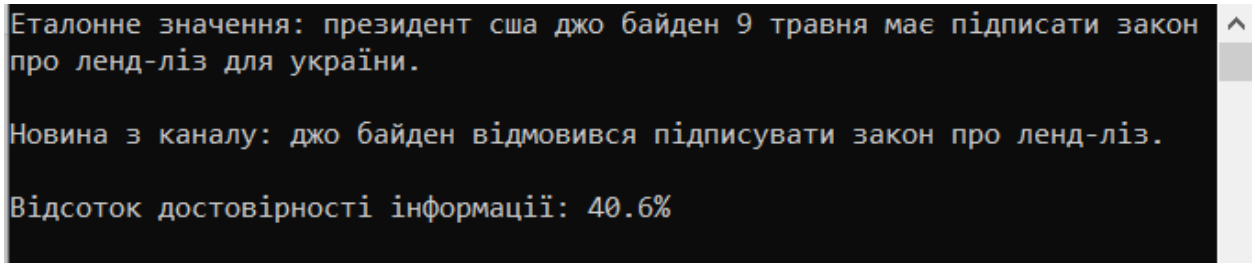


Рисунок 4.5 – Результат системи з використанням модифікованого методу Дамерау-Левенштейна

Аналіз часу витрачений на алгоритм наведений у таблиці 4.4.

Таблиця 4.4 – Час витрачений на обробку даних

Метод	Час
Метод Дамерау-Левенштейна	694 мкс.
Модифікований метод Дамерау-Левенштейна	725 мкс

На рисунку 4.6 зображено графік залежності відсотків подібності текстів та валідності значень.

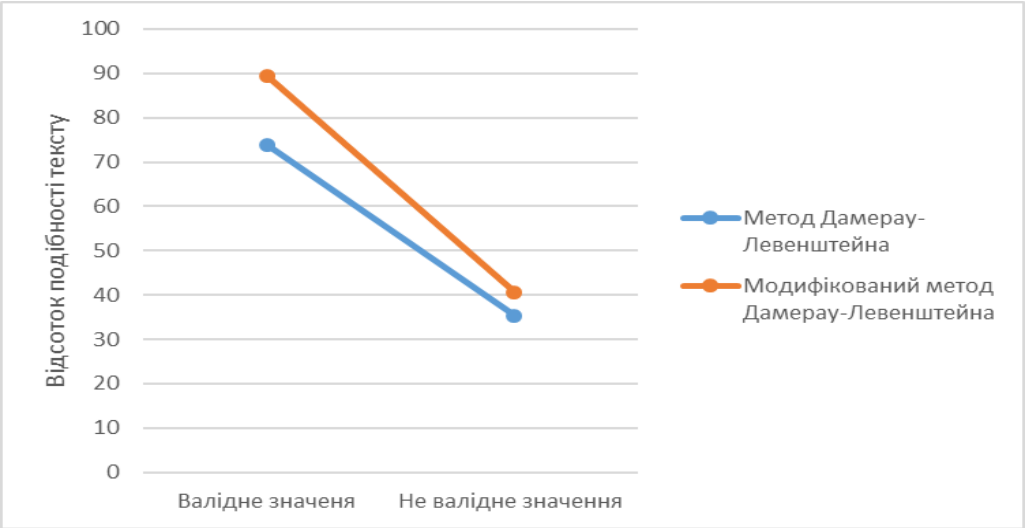


Рисунок 4.6 – Графік залежності відсотків подібності тексту

На рисунку 4.7 зображено графік залежності часу витраченого на обробку тексту та валідності значень.

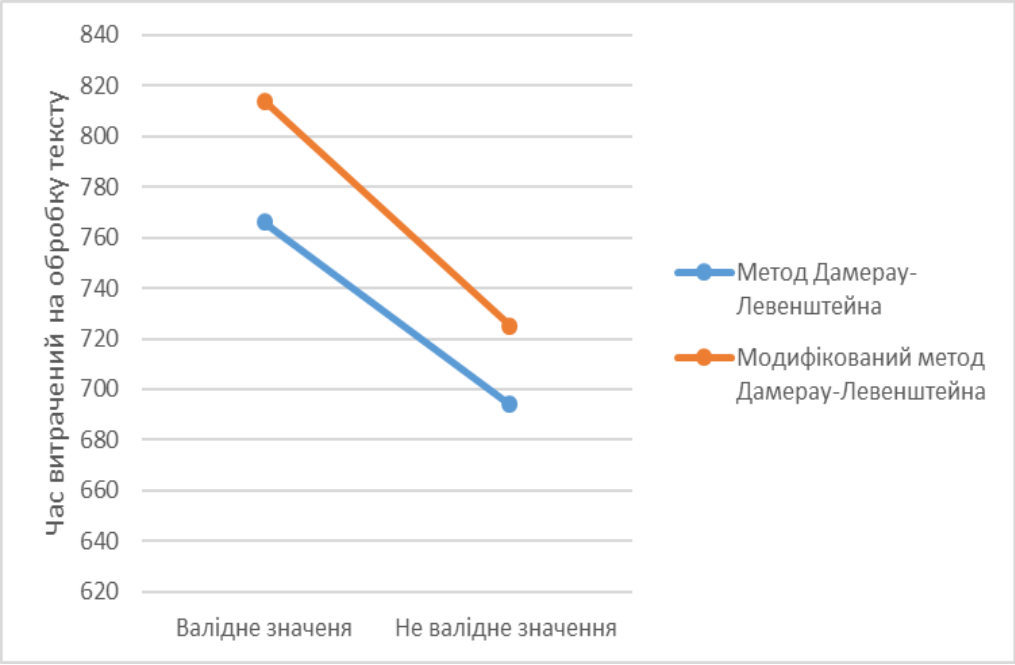


Рисунок 4.7 – Час витрачений на обробку тексту

З отриманих результатів можна зробити висновок, що модифікований метод Дамерау-Левенштейна дає більш точний відсоток близькості тексту але витрачає більше часу на обробку вхідних даних. Відсоток близькості тексту зменшився на 5-15%, в залежності від кількості вхідної інформації. Час затрачений на обробку даних збільшився на 5%.

4.2 Удосконалення системи

Для удосконалення системи запропоновано розпаралелити обробку вхідної текстової інформації та обчислення коефіцієнту близькості тексту. Також на цій технології можна реалізувати Text parsing сервер, який зчитує та аналізує текст з офіційних джерел.

Розглядається один з підходів скорочення часу обчислень алгоритмів, що добре розпаралелюються, заснований на спільному використанні класичних і графічних процесорів. Для цього розглядається CUDA-X AI від компанії NVIDIA. CUDA-X AI забезпечує кращу в світі продуктивність, а також дозволяє розробляти програми глибокого навчання на настільних комп’ютерах або розгортати їх в центрах обробки даних.

ВИСНОВКИ

Обробка природної мови завжди була актуальною, оскільки велика частина даних, що надходять від користувача, знаходиться в неструктурованій формі, наприклад у вигляді вільного тексту, будь то коментарі користувачів (Facebook, Instagram), чати (Whatsapp, Hike), огляди продуктів (Amazon, Flipkart) і т.д.

Щоб комп'ютер розумів будь-який текст, потрібно розбити це слово так, щоб машина могла його зрозуміти. Ось тут і з'являється концепція токенізації в NLP. Токенізація грає важливу роль в роботі з текстовими даними і має завдання розбиття послідовності тексту на блоки з семантичним значенням. Ці одиниці називаються токенами, і складність токенізації полягає в тому, що для отримання ідеального поділу, треба щоб всі токени в тексті мали правильне значення і не було пропущених токенів.

Токенізація — фундаментальний крок, як в традиційних методах NLP, таких як Count Vectorizer, так і в архітектурі на основі розширеного глибокого навчання, таких як Transformers [8].

Семантична близькість термів вже давно є невід'ємною частиною теорії обробки текстів природною мовою. Семантична близькість між двома сутностями з часом може змінюватися у зв'язку зі змінами словників. Крім того, семантична близькість двох сутностей може бути різною в різних предметних областях. Також важливою частиною виступає виправлення слів.

Виправлення слів використовується для пошуку неправильного слова у письмовій формі. Відстань Левенштейна - один з алгоритмів виправлення друкарської помилки. Це алгоритм, який обчислює різницю між двома рядками. Операції, що використовуються для обчислення, - це вставка, видалення та заміна. Однак цей алгоритм має недолік, який полягає в тому, що він не може подолати дві літери, що перемикаються, в тому самому слові. Алгоритм, який може вирішити ці проблеми, це алгоритм Дамерау

Левенштейна. Метод може обчислити менші відстані редагування, якщо суміжні операції транспонування можуть замінити кілька операцій заміни. Ефективні реалізації також ґрунтуються на динамічному програмуванні.

Для модифікації методу був використаний метод стемінгу слова. Завдяки цьому кожне слово обробляється функцією, яка видаляє закінчення та префікси у слові. Таким чином в методі Дамерау-Левенштейна є можливість не використовувати операції вставки та перестановки символів. Це впливає на значення відстані між словами, що дає можливість чітко визначати однакові слова. Отже для обробки слова кращім є цей метод для виключення орфографічних помилок.

З тестування було виявлено, що модифікований метод Дамерау-Левенштейна краще аналізує слова з орфографічними помилками, але витрачає більше чану на обробку слова.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Барковська О. Ю., Ляшова А. О. АКТУАЛЬНІСТЬ МЕТОДІВ ВИЗНАЧЕННЯ ТЕКСТОВОЇ БЛИЗЬКОСТІ ДЛЯ АНАЛІЗУ ПУБЛІКАЦІЙ У НОВИННИХ КАНАЛАХ [Електронний ресурс] / Барковська О. Ю., Ляшова А. О. // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. – 2022. – Режим доступу до ресурсу: https://nure.ua/wp-content/uploads/conf-2022-akov/telecom_2022_volume_1.pdf.

2) Що таке контроль версій? [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/ru/git/tutorials/what-is-version-control.

3) Що таке VCS (система контролю версій) [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://habr.com/ru/post/552872/.

4) Системи перекладу та розпізнавання тексту [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://obrazovaka.ru/informatika/sistemy-perevoda-i-raspoznavaniya-teksta.html.

5) Система розпізнавання зображень [Електронний ресурс] – Режим доступу до ресурсу: https://polygant.net/ru/ai/sistema-raspoznavaniya-izobrazhenij/.

6) Git distributed is the new centralized [Електронний ресурс] – Режим доступу до ресурсу: https://git-scm.com/.

7) Що таке Git? [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/ru/git/tutorials/what-is-git.

8) Перекладач Google [Електронний ресурс] – Режим доступу до ресурсу: https://startpack.ru/application/google-translate#features.

9) Що таке обробка природної мови та як вона працює? [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://uk.omatomeloanhikaku.com/what-is-natural-language-processing-and-how-does-it-work-6175.

10) Плавне введення у Natural Language Processing (NLP) [Електронний

ресурс] – Режим доступу до ресурсу: https://datastart.ru/blog/read/plavnoe-vvedenie-v-natural-language-processing-nlp.

11) Diego Lopez Yse. Your Guide to Natural Language Processing (NLP) [Електронний ресурс] / Diego Lopez Yse – Режим доступу до ресурсу: https://www.datasciencecentral.com/profiles/blogs/your-guide-to-natural-language-processing-nlp

12) Ane Berasategi. Overview of tokenization algorithms in NLP [Електронний ресурс] / Ane Berasategi – Режим доступу до ресурсу: https://towardsdatascience.com/overview-of-nlp-tokenization-algorithms-c41a7d5ec4f9.

13) ARAVIND PAI. What is Tokenization in NLP? Here’s All You Need To Know [Електронний ресурс] / ARAVIND PAI. – 2020. – Режим доступу до ресурсу:https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/.

14) Про метод визначення текстової близькості заснований на семантичних класах [Електронний ресурс] / С. Х. Г. Бермудес, С. У. Керимова – Режим доступу до ресурсу: http://www.ivdon.ru/uploads/article/pdf/IVD_65_Bermudez_Kerimova__2_.pdf_1a8adc1b77.pdf.

15) Метод порівняння текстів для розв'язання пошуково-аналітичних задач [Електронний ресурс] / И.В. Соченков. – 2013. – Режим доступу до ресурсу: http://www.isa.ru/aidt/images/documents/2013-02/32_43.pdf.

16) Визначення семантичного близькості термів з допомоги контекстного множества / Бондарчук Д.В., 2015.

17) Огляд засобів латентно-семантичного аналізу [Електронний ресурс] / Дмитро Геннадійович Медведєв* , Владислав Миколайович Пірогов≠ // Кафедра інформатики та прикладної математики, Криворізький державний педагогічний університет, пр. Гагаріна, 54, м. Кривий Ріг, 50086, Україна – Режим доступу до ресурсу: https://lib.iitta.gov.ua/706571/1/600-1-2639-1-10-20170502.pdf.

18) Семантичний аналіз подоби текстів [Електронний ресурс] / Смехун Я.А. – Режим доступу до ресурсу: https://research-journal.org/technical

/semanticheskij-analiz-podobiya-tekstov/.

19) String correction using the Damerau-Levenshtein distance [Електронний ресурс] / Chunchun Zhao, Sartaj Sahni. – 2019. – Режим доступу до ресурсу: https://bmcbioinformatics.biomedcentral.com/articles /10.1186/s12859-019-2819-0.

20) Python text analysis tools: Levenshtein Distance [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.aylakhan.tech/?p=736.