НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

«На правах рукопису»
УДК 004.912

До захисту допущено:
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою

«Системне програмування і спеціалізовані комп'ютерні системи»

зі спеціальності 123 «Комп'ютерна інженерія»

на тему: «Засоби формування та обробки бази даних словосполучень української мови**»**

Виконала:
студентка ІІ курсу, групи КВ-01мп
Рябоконь Тетяна Олексіївна _____

Науковий керівник:
Доцент кафедри СПСКС, к.т.н, доцент,
Петрашенко Андрій Васильович _____
_____

Рецензент:
Посада, науковий ступінь, вчене звання,
Прізвище, ім'я, по батькові _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – другий (магістерський)

Спеціальність – 123 «Комп‘ютерна інженерія»

Освітньо-професійна програма «Системне програмування і спеціалізовані комп‘ютерні системи»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___»_____2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Рябоконь Тетяні Олексіївні

1. Тема дисертації «Засоби формування та обробки бази даних словосполучень української мови», науковий керівник дисертації Петрашенко Андрій Васильович, к.т.н, доцент, затверджені наказом по університету від «5» 11. 2021 р. №3682-С

2. Термін подання студентом дисертації: 7.12.2021

3. Об’єкт дослідження: методи формування, прискорення та оброблення бази даних словосполучень української мови

4. Вихідні дані:
   - розроблена програмна система, що дозволяє наповнювати базу даних словосполучень, шляхом оброблення текстових даних;
   - визначено які статистичні методи найкраще підходять для виділення словосполучень з текстів саме українською мовою.

5. Перелік завдань, які потрібно розробити
   - дослідити існуючи методи пошуку словосполучень у текстових даних;
   - на основі експерименту обрати метод пошуку словосполучень, який підходить для роботи з текстами саме українською мовою;
   - обрати критерії оцінки методів та порівняти методи за допомогою експерименту;
   - розробити програмну систему для формування бази даних словосполучень;
   - дослідити можливі методи прискорення оброблення текстових даних у розробленій системі;

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
   - схема архітектури системи;

- схема модулю виділення словосполучень з текстових даних
- графіки влучності та повноти для словосполучень з двох та трьох слів знайдених в ході експерименту
- графік розподілу за кількістю унікальних слів у реченнях до та після нормалізації.

7. Орієнтовний перелік публікацій:
- тези доповіді на конференції ПМК 2021 «Програмні засоби формування та обробки бази даних словосполучень української мови»,
- стаття на в науковому журналі "Комп'ютерно-інтегровані технології: освіта, наука, виробництво", випуск № 44 «Програмні засоби формування та обробки бази даних словосполучень української мови»

9. Дата видачі завдання 7.10.2020

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	10.12.2020	
2.	Визначення структури магістерської дисертації та вивчення літератури, пошук додаткової літератури.	30.01.2021	
3.	Робота над першим розділом магістерської дисертації та проведення наукового дослідження.	15.04.2021	
4.	Проведення наукового дослідження. Робота над другим розділом магістерської дисертації. Розроблення програмного забезпечення. Підготовка матеріалів доповіді ПМК-2021.	15.09.2021	
5.	Проведення наукового дослідження. Робота над статтею за результатами наукового дослідження.	10.10.2021	
6.	Проведення наукового дослідження. Робота над третім розділом магістерської дисертації.	31.10.2021	
7.	Завершення роботи над основною частиною магістерської дисертації. Підготовка ілюстративного матеріалу.	20.11.2021	
8.	Попередній розгляд магістерської дисертації на кафедрі - день передзахисту	01.12.2021	

Студент Тетяна РЯБОКОНЬ

Науковий керівник Андрій ПЕТРАШЕНКО

РЕФЕРАТ

Актуальність теми. Аналіз словосполучень є важливим розділом NLP досліджень. Вміння аналізувати, класифікувати та знаходити словосполучення дає можливість оперувати контекстом та змістом, що закладені у речення, а не окремими словами. Це допомагає значно вдосконалити системи, що працюють з натуральними мовами. Існує багато алгоритмів та підходів, які дозволяють аналізувати окремі слова, але аналіз та пошук групи слів, що зв’язані між собою є більш складним завданням.

Словосполучення важливі для ряду застосувань: генерація природної мови – щоб переконатися, що вихідні дані звучать природно і уникнути помилок; обчислювальна лексикографія – для автоматичного визначення важливих словосполучень, які мають потрапити до словника та корпусні лінгвістичні дослідження, наприклад, вивчення суспільних та культурних явищ через мову.

Дана робота присвячена пошуку словосполучень в текстах, що написані українською мовою, з подальшим упорядкуванням та морфологічним аналізом слів. Виділення словосполучення - це задача, що передбачає використання комп'ютера для автоматичного виділення словосполучення з корпусу. Традиційний метод виконання виділення словосполучення полягає у знаходженні формули на основі статистичних величин для обчислення оцінки пов’язаної з кожною парою слів.

Мета роботи: підвищення ефективності автоматичної генерації бази даних словосполучень української мови, а також розроблення засобів пошуку.

Об’єктом дослідження є тексто-орієнтовані бази даних.

Предметом дослідження є методи та алгоритми автоматизованого генерування бази даних словосполучень за допомогою оброблення текстових даних, а також методи прискорення генерування описаної бази даних.

Методи дослідження в роботі використовуються статистичні міри асоціації, методи нормалізації текстових даних та методи розподілених обчислень та оброблення даних.

Наукова новизна: розроблено програмні засоби формування бази даних словосполучень української мови, яка відрізняється від існуючих тим, що ефективність генерації розробленої бази даних підвищена за допомогою методів розподілених обчислень. Розроблене програмне забезпечення дозволяє знаходити словосполучення шляхом оброблення текстів українською мовою та зберігати їх до сховища даних, з можливістю пошуку по цим даним та їх подальшого аналізу.

Практична цінність отриманих в роботі результатів полягає в тому, що розроблена база даних може використовуватися для подальшого вивчення мови та інших досліджень у сфері NLP. Також проведено дослідження та отримані дані, які методи пошуку словосполучень найкраще підходять саме для української мови, враховуючи граматичні особливості мови, що може бути використано у подальших дослідженнях генерації природної мови, для вдосконалення пошукових систем та голосових асистентів, інструментів, що домагають редагувати та підсумовувати тексти, тощо.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на XVI науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2021. Стаття у науковому журналі "Комп'ютерно-інтегровані технології: освіта, наука, виробництво", випуск № 44 «Програмні засоби формування та обробки бази даних словосполучень української мови»

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів та висновків.

У вступі подано загальну характеристику дослідження, розглянуто актуальність та новизну дослідження, названі можливі практичні застосування розроблюваної бази даних словосполучень.

У першому розділі проаналізовано існуючі подібні системи, названі їх переваги та недоліки, визначено вимоги до розроблюваної системи та сформульовано проблему попередньої оброки текстових даних.

У другому розділі описано алгоритм попередньої обробки текстів, наведено результати аналізу методів знаходження словосполучень у текстах українською мовою, порівняно їх та обрано найефективніші, та проаналізовано способи фільтрації знайдених словосполучень.

У третьому розділі описано інструменти використані для розроблення програмного забезпечення, архітектуру програмного забезпечення та засоби прискорення генерації бази даних словосполучень.

У четвертому розділі визначено можливу сферу застосування та наведено результати оптимізації генерації розробленої бази даних словосполучень, а також результати роботи розробленого програмного забезпечення.

У висновках представлені результати проведеної роботи.

Робота представлена на 93 аркушах, містить посилання на список використаних літературних джерел.

Ключові слова: статистичні методи знаходження словосполучень, база даних словосполучень, словосполучення, текстовий корпус.

ABSTRACT

Actuality of theme. Collocation analysis is an important part of NLP research. The ability to analyze, classify and find collocations makes it possible to operate with the context and content embedded in sentences, rather than individual words. This helps to significantly improve systems that work with natural languages. There are many algorithms and approaches that allow you to analyze individual words, but analyzing and finding a group of related words is a more difficult task.

Collocations are important for such applications as: natural language generation - to make sure that the source data sounds natural and to avoid mistakes; computational lexicography - to automatically identify important phrases to be included in the dictionary and corpus linguistic research, such as the study of social and cultural phenomena through language.

This work is devoted to collocations extraction from texts written in the Ukrainian language, filtering and morphological analysis of words. Collocation extraction is a task that involves using a computer to automatically select a collocation from the text corpus. The traditional method of collocation extraction is to find a formula based on statistical values to calculate the score associated with each pair of words.

Purpose: to increase the efficiency of automatic generation of the database of phrases of the Ukrainian language, as well as the development of search tools.

Object of research is text-oriented databases.

Subject of research is methods and algorithms of automated generation of a database of phrases by means of text data processing, and also methods of acceleration of generation of the described database.

Methods of research: the study uses statistical measures of association, methods of normalization of text data and methods of distributed computing and data processing.

Scientific novelty: software tools for the generation of the Ukrainian language database of collocation have been developed. It differs from the existing ones in that the efficiency of generating the developed database has been increased with the help of distributed computing methods. The developed software allows finding collocations by

processing texts in the Ukrainian language and saving them to the data storage, for searching through this data and their further analysis.

Practical value of the results obtained in this work is that the developed database can be used for further language study and other research in the field of NLP. Research was also conducted and data were obtained on which collocation extraction methods are best suited for the Ukrainian language, taking into account the language grammatical features that can be used in further studies of natural language generation, to improve search engines and voice assistants, tools for editing and summarizing texts, etc.

Approbation. The main provisions and results of the work were presented and discussed at the XVI scientific conference of undergraduates and graduate students "Applied Mathematics and Computing" AMC-2021. Article in the scientific journal "Computer-integrated technologies: education, science, production", issue № 44 "The software system for generation and processing of a database of collocations of the Ukrainian language".

Structure and scope of work. The master's dissertation consists of an introduction, four chapters and conclusions.

In the introduction the general characteristic of research is given, urgency and novelty of research are considered, possible practical applications of the developed database of phrases are named.

The first section analyzes the existing similar systems, names their advantages and disadvantages, identifies the requirements for the developed system and formulates the problem of preliminary processing of text data.

The second section describes the algorithm of pre-processing of texts, presents the results of analysis of methods for finding collocation in texts in Ukrainian, compares them and selects the most effective, and analyzes ways to filter the found phrases.

The third section describes the tools used to develop the software, the software architecture, and the tools for accelerating the generation of the phrase database.

The fourth section identifies the possible scope and presents the results of optimizing the generation of the developed database of phrases, as well as the results of the developed software.

The conclusions present the results of the work.

The work is presented on 93 sheets, contains references to the list of used literature sources.

Keywords: statistical methods of finding phrases, database of phrases, colocation, text corpus.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Колокація — у корпусній лінгвістиці це ряд слів або термінів, які зустрічаються разом частіше, ніж випадково.

N-грам — в обчислювальній лінгвістиці це безперервна послідовність з n елементів із заданого зразка тексту або мовлення. Елементами можуть бути фонеми, склади, букви, слова або пари основ відповідно до призначення.

Обробка природної мови (англ. Natural-language processing, NLP) — це розділ лінгвістики, інформатики та штучного інтелекту, який займається взаємодією між комп'ютерами та людською мовою, зокрема, вивчає як програмувати комп'ютери для оброблення та аналізу великих обсягів даних природної мови.

Текстовий корпус — це мовний ресурс, що складається з великого і структурованого набору текстів, зазвичай, зберігаються та обробляються в електронному вигляді.

Стеммінг — це процес зведення похідних слів, або слів утворених словозаміною, до їх основи чи кореня — як правило, письмової форми слова.

Лемматизація — це процес групування форм слова утворених словозаміною, з метою аналізу як одного елемента, ідентифікованого лемою слова або словниковою формою.

Стоп-слова — це будь-яке слово у стоп-листі, які відфільтровуються до або після обробки текстових даних. Не існує єдиного універсального списку стоп-слів, які використовуються всіма інструментами обробки природної мови, ані узгоджених правил визначення стоп-слів, насправді не всі інструменти навіть використовують такий список.

Влучність та повнота — це показники ефективності, які застосовуються до даних, отриманих із колекції, корпусу або простору вибірки.

POS- тегування або розмічування за частинами мови — це процес позначення слова в тексті або корпусі як відповідного певній частині мови.

Токенізація — це процес розмежування та, можливо, класифікації розділів вхідних текстових даних.

Прикладний програмний інтерфейс (англ. Application Programming Interface, API) — це інтерфейс, який забезпечує контракт зв’язку між комп’ютерами або між комп’ютерними програмами.

Spark DataFrame — це розподілена колекція даних, організована в іменовані стовпці, концептуально схожа на таблицю в реляційній базі даних.

Spark Resilient Distributed Dataset (RDD) — стійкий розподілений набір даних, який представляє собою набір елементів, розділених між різними вузлами кластера, з якими можна працювати паралельно.

JavaScript Object Notation (JSON) — це відкритий стандарт формату обміну даними, який використовує зрозумілий людині текст для зберігання та передачі об’єктів даних, що складаються з пар атрибут-значення та масивів або інших значень, які можна серіалізувати.

ВСТУП

На сьогоднішній день існує велика кількість різних способів вивчення та дослідження людської природної мови. Досліджувати мову потрібно не тільки для отримання нових лінгвістичних знань, а й для розвитку інформаційних технологій. Найвідоміший зі способів вивчення природної мови за допомогою комп'ютера – обробка природної мови.

Обробка природної мови – це наукова дисципліна та напрям в комп'ютерних науках, який поєднує галузі штучного інтелекту та лінгвістики. Ця дисципліна використовує лінгвістичні знання, а саме граматику, задля вивчення проблем автоматизованого генерування та розуміння природних людських мов, видобутку та пошуку даних, машинного перекладу та багатьох інших. Один з підходів до обробки природної мови – це статистичний підхід. Такий підхід використовує стохастичні, ймовірнісні та статистичні методи для вирішення деяких проблем, з якими стикається традиційне NLP. Наприклад, у статистичному NLP використанням алгоритмів машинного навчання та великих корпусів досягається однозначність результатів аналізу великих речень та відривків тексту.

Одне з багатьох завдань статистичного NLP – це автоматичне вилучення словосполучень з тексту і воно передбачає пошук словосполучень у великих текстових корпусах. Словосполучення мають багато можливих застосувань: пошук зв'язків між словами у тексті для цілей індексування під час пошуку інформації, автоматична генерація мови, багатозначність слова в багатомовній лексикографії, удосконалення систем категоризації тексту тощо.

Існує також багато ресурсів для вивчення та дослідження мови, наприклад, словники, перекладачі, текстові корпуси та майже будь-які тексти. Через те що мова трансформується та змінюється, для точної та ефективної роботи комп'ютерних алгоритмів, що працюють з живою мовою, потрібен аналіз справжнього мовлення. Як вже було сказано, одним з видів мовних ресурсів є словники різної тематики та призначення, у тому числі й словники словосполучень.

Дана робота присвячена дослідженню методів автоматичного формування словника словосполучень української мови за допомогою розроблення програмної

системи автоматичного аналізу текстів для знаходження в них словосполучень, розробленню ефективних алгоритмів генерації такого словника, фільтрації результатів аналізу, нормалізації текстів та розробленню засобів пошуку результатів для зручного користування створеним словником.

1. АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Огляд задачі виділення словосполучень з тексту

У наш час інтерес до досліджень словосполучень пов'язаний з розробленням нових пошукових систем та систем машинного перекладу, технологій розпізнавання тексту та рекомендаційних систем. Важливим завданням сучасної корпусної лінгвістики є вилучення відповідної лінгвістичної інформації, зокрема за допомогою використання статистичних методів. Крім того, корпусні дослідження дозволяють перевірити лінгвістичні теорії та гіпотези, а також виявити та інтерпретувати нові мовні факти.

В лінгвістиці поняття словосполучення визначають як вираз, що складається з двох або більше слів, що відповідають певному загальноприйнятому способу висловлювання. Іншими словами, словосполучення – це синтаксична одиниця утворена з декількох слів, що поєднані між собою за допомогою підрядного зв’язку. Словосполучення класифікують за будовою та за способом вираження головного слова. За будовою вони поділяються на прості та складні, тоді як за способом вираження головного слова вони бувають:

- іменні – іменникові, прикметникові та займенникові;
- дієслівні,
- прислівникові.

Словосполучення характеризуються обмеженою композиційністю. Вираз природної мови ми називаємо композиційним, якщо значення виразу можна передбачити за значенням частин, з яких він складається. Словосполучення не є повністю композиційними, оскільки до комбінації зазвичай додається елемент значення даного виразу. У випадку словосполучення «пшеничне волосся» слово «пшеничне» вказує на те, що волосся має колір пшеничних колосків, але не є з пшениці, тобто слово вжите в переносному значенні, а отже без контексту не можна дізнатися значення даного словосполучення. В той же час не можуть бути словосполученнями однорідні члени речення, поєднання службових частин мови і

сполучення підмета та присудка, хоча такі мовні конструкції можуть часто зустрічатися у тексті разом. Фразеологізми – це найяскравіші приклади некомпозиційності. Через цю властивість аналіз словосполучень комп'ютерними алгоритмами є нетривіальним завданням.

Узгодженість слів є важливим чинником у переважній більшості завдань з опрацювання природної мови та моделювання мови, а саме: синтаксичного розбору, машинного перекладу, генерації мовлення, автоматичного та напівавтоматичного наповнення словників, визначення семантичних ролей, тощо.

Термін колокація має як лінгвістичний, так і лексикографічний характер. Він має різні визначення, але жодне з них не є загальноприйнятим. Розглянемо визначення з роботи Я. Чуека (1988), який визначає колокаційний вираз як «синтаксичну та семантичну одиницю, чиє точне й однозначне значення чи побіжне значення не може бути отримано безпосередньо із значення чи побіжних значень його компонентів». Це поняття колокації є відносно широким і охоплює широкий спектр лексичних явищ, таких як ідіоми, фразові дієслова, сполучення легких дієслів, технологічні вирази, власні назви та стандартні фрази. [1]

Феномен словосполучень привернув увагу лінгвістичної спільноти, коли Ферт (1957) визначив його як "слова, які часто зустрічаються разом", або "компанія, яку тримають слова". Хоча сучасні технології значно зменшили потребу покладатися виключно на інтуїцію, на момент, коли Ферт сформулював своє визначення, інтуїція лінгвіста була єдиним, що використовувалося для ідентифікації словосполучень. [2] Взявши за основу висновки Ферта, з часом дослідники корпусної лінгвістики розробили кількісне визначення словосполучення. Наприклад, М. Хой визначив поняття словосполучення як «зв'язок, який має лексична одиниця, з елементами, які з'являються з більшою, ніж випадковою ймовірністю, в текстовому контексті". [3]

Багато років дослідники намагалися дати визначення словосполученням, або як вони ще називаються – колокаціям, але навіть сьогодні не існує одного загальноприйнятого визначення.

Однак існують три критерії, які задовольняє більшість колокацій:

- некомпозиційність – означає, що значення всієї колокації – це більше, ніж сума значень слів, що її утворюють.
- незамінність – означає, що ми не можемо замінити слово в колокації іншим словом, що має подібне чи навіть таке саме значення.
- незмінність – означає, що ми не можемо вільно змінювати колокацію додатковим лексичним матеріалом або переносити колокацію через деякі граматичні перетворення, що особливо справедливо для фразеологізмів.

Дослідження словосполучень важливі для ряду застосувань в інформатиці:

- генерація природної мови – щоб переконатися, що вихідні дані звучать природно і уникнути помилок;
- обчислювальна лексикографія – для автоматичного визначення важливих словосполучень, які мають бути внесені до словника;
- синтаксичний розбір – таким чином перевага може бути надана фразам, в яких слова більш природно пов’язані між собою;
- корпусні лінгвістичні дослідження – наприклад, вивчення суспільних та культурних явищ через природну мову .

В даній роботі також буде приділена увага статистичним методам для вилучення словосполучень з текстових корпусів, таким як: вибір словосполучень за частотою зустрічальності, вибір на основі середнього значення та дисперсії відстані між головним та залежним словом, що включає в себе, методи перевірки гіпотез та метод взаємної інформації. Також буде визначений метод, що працює найкраще саме з текстами українською мовою. Іншим не менш важливим напрямком дослідження в рамках цієї роботи є фільтрація знайдених статистичними методами колокацій.

1.2. Аналіз основних методів виділення словосполучень

Загалом, пошук колокацій є найпопулярнішим застосуванням мір лексичних асоціацій, і на цю тему опубліковано досить багато значущих досліджень. У

комп’ютерній лексикографії автоматична ідентифікація колокацій використовується, щоб допомогти лексикографам у складанні лексикографічної інформації (визначення можливих значень слів, лексичних переваг, прикладів використання тощо) для традиційної лексики або для спеціальної лексики ідіом та колокацій, що використовуються, наприклад, у перекладах, двомовних словниках або для викладання мови.

Колокації відіграють важливу роль у системах генерації природної мови, де під час добору слів використовуються колокації та ідіоми, щоб покращити плавність автоматично генерованого тексту.

Міри лексичної асоціації — це математичні формули, що визначають силу асоціації між двома чи більше словами на основі їх входження та співвходжень у текстовому корпусі. Вони мають широкий спектр застосувань у сфері обробки природної мови та обчислювальної лінгвістики, таких як автоматичне виділення колокацій, двомовне вирівнювання слів або розбір залежностей. [4]

Протягом останніх десятиліть було введено ряд різноманітних асоціаційних заходів. [5] Декілька дослідників також спробували порівняти існуючі методи та запропонувати різні схеми оцінювання, наприклад, К. Кіта (1994) та С. Еверт (2001). [6, 7]

Для знаходження колокацій в текстових корпусах були запропоновані різні статистичні методи. Їхня реалізація стало можлива завдяки появі великої кількості електронних мовних ресурсів та збільшенню потужності комп’ютерів, що дозволило швидше аналізувати велику кількість текстів. Серед таких методів можна виділити: t-критерій Стьюдента, метод точкової взаємної інформації, логарифмічна правдоподібність, критерій хі-квадрат та метод простого підрахунку.

Велика кількість наукових праць та підручників загальної статистики містять відомості та дослідження про t-критерій Стьюдента, одна з найвідоміших таких праць – це «Статистичні методи» Дж. Снедекора та В. Кокрена. [8]

Інша відома публікація, що також була присвячена дослідженню колокацій – це робота К. Черча та П. Хенкса (1989), в якій автори доходять висновку, що метод взаємної інформації є ефективним для ідентифікації колокацій в лексикографії.

Також авторами був створений новий тип словника, що побудований на основі корпусу. Ними також був розроблений підхід в обчислюваній лексикографії, який поєднує обчислювальні методи, корпусні дані та людське сприйняття мови. Це дозволило створювати більш точні словники, які краще відображають реальне використання мови. [9]

В цей же час Т. Данінг (1993) в своїй роботі вказує на низьку ефективність методу взаємної інформації. Згідно з його дослідженнями, метод перевірки відношенням правдоподібностей є більш ефективним для виявлення одномовних словосполучень, особливо якщо їхня кількість у тексті не дуже велика. [10]

В роботі К. Маннинга та Г. Шютце (1999) описані основні статистичні методи, їхні переваги та недоліки. Автори доходять висновку, що найефективніша стратегія пошуку словосполучень в текстовому корпусі – це поєднання статистичної та лінгвістичної моделі. [5]

Ф. Смаджа (1993) розробив алгоритм, який називається Xtract. Перший етап цього алгоритму представляє собою виділення груп слів на основі лише статистичної інформації, наступні етапи – виділення стійких словосполучень та створення шаблонних виразів на основі біграм, а також встановлення для них синтаксичних відношень. [11]

Ще одна робота Ф. Смаджа в співавторстві з К. Макк'юєн зосереджена на виділенні словосполучень більшої довжини та фразеологізми як вирішення завдання генерації мовлення. Для цього вони використовують багато статистичних даних (частоти слів, відхилення, відстані тощо). [12]

У дослідженні Дж. П. Голдман і Е. Верлі була використана систему FipsCo для виділення термінів, тому вони покладаються на дуже потужний синтаксичний аналізатор. [13] На відміну від них, Ву і Чанг намагалися витягти колокації з двомовного вирівняного корпусу, і для цього вони використали ряд кроків попередньої обробки в поєднанні з коефіцієнтом логарифмічної правдоподібності та алгоритмом вирівнювання слів, тоді як О. Вечтомова використовує довгі колокації для розширення запиту при інформаційному пошуку. [14, 15]

Для порівняння мір асоціації необхідно визначити критерій, структуру для їх оцінки. На жаль, не існує єдиного загально прийнятого критерію для оцінки мір асоціації, з яким погоджується більшість дослідників, але існує ряд різних підходів, які використовуються різними авторами. Наприклад, Ф. Смаджа використав навички професійного лексикографа, щоб вручну позначати n-грами як колокації або неколокації. [12] А. Танопулос і Пірс використовують систему WordNet як стандарт. [16, 17] Тоді як Еверт і Кренн використовують для порівняння невелику випадкову вибірку з усього набору кандидатів. [7] Десь посередині лежить підхід да Сілви та Лопеса. [18] Вони вручну перевірили кілька сотень випадково вибраних n-грам із набору, отриманого кожним із перевірених показників, позначили їх як колокації та неколокації, і на основі цього обчислили точність. Кожен з цих методів має свої переваги та недоліки — підхід Смаджі дає дуже точне значення для влучності та повноти, але, з іншого боку, займає дуже багато часу. Метод Танопулоса є швидшим, але, він сам стверджує, що WordNet не є точним та повним щодо некомпозиційних колокацій. В той час як метод Еверта є найшвидшим і добре працює для визначення ефективності мір асоціації, але можна лише передбачити справжні результати вимірювання влучності та повноти для міри асоціації. Довірчі інтервали для оцінки залежатимуть від розміру випадкової вибірки. За допомогою методу, який використовували да Сілва і Лопес, неможливо обчислити влучність та повноту, тому вони використовують загальну кількість багатослівних одиниць, знайдених кожною мірою, як непрямий показник.

Тема знаходження колокацій у текстових даних знайшла відклик у багатьох наукових працях, отже, можна зробити висновок, що проблема сумісності слів або виділення словосполучень з текстових даних є дійсно актуальною та активно досліджуваною проблемою у корпусній лінгвістиці.

1.3. Огляд існуючих систем знаходження словосполучень

Частиною розробленого словника словосполучень, що буде оновлюватися за допомогою аналізу текстових корпусів, є система автоматичного виділення

словосполучень з текстів. Існують аналоги такої системи, наприклад, Sketch Engine, WordSmith Tools 4 (WST 4), Collocate та Ngram Statistics Package (NSP). Деякі з цих програмних пакетів є повноцінними конкордансерами, а інші спеціалізуються виключно на виділенні колокації.

1.3.1. Sketch Engine

Sketch Engine - це програмне забезпечення для аналізу текстів та керування текстовим корпусом, розроблене компанією Lexical Computing Limited. Його основне призначення – визначати, що є типовим та частим для використання у мові, а що рідкісним, застарілим, що виходить з ужитку або які нові слова чи граматичні конструкції починають використовуватися.

За пошук словосполучень відповідає частина даної системи під назвою Word Sketch. Система обробляє слова, що зв'язані з шуканим словом, та інші слова в його оточенні, надає інформацію про граматичну та розмовну поведінку слова. Результати аналізу впорядковуються за категоріями, які називаються граматичними відносинами. Те які слова будуть включені до аналізу, визначається правилами, що записані в системі, яка виявляє потенційні колокації. Ці правила залежать від мови, якою написані тексти, що аналізуються. Також правила вони можуть бути змінені користувачем.

За замовчуванням Word Sketch сортує словосполучення, за частотою. Це найкращий варіант для більшості випадків використання, тому що те, що часто зустрічається, зазвичай є типовим. Індекс Соренсена, t-критерій та взаємна інформація використовуються для визначення того, наскільки типовою (або наскільки сильною) є колокація. Велике значення оцінки означає, що колокація є сильною, а якщо значення низьке – колокація відповідно слабка.

Але все ж основна мета оцінки - відсортувати колокації за їх типовістю чи силою, а не вирішувати, чи є вони слабкими чи сильними.

Приклад роботи Word Sketch можна побачити на Рисунку 1.

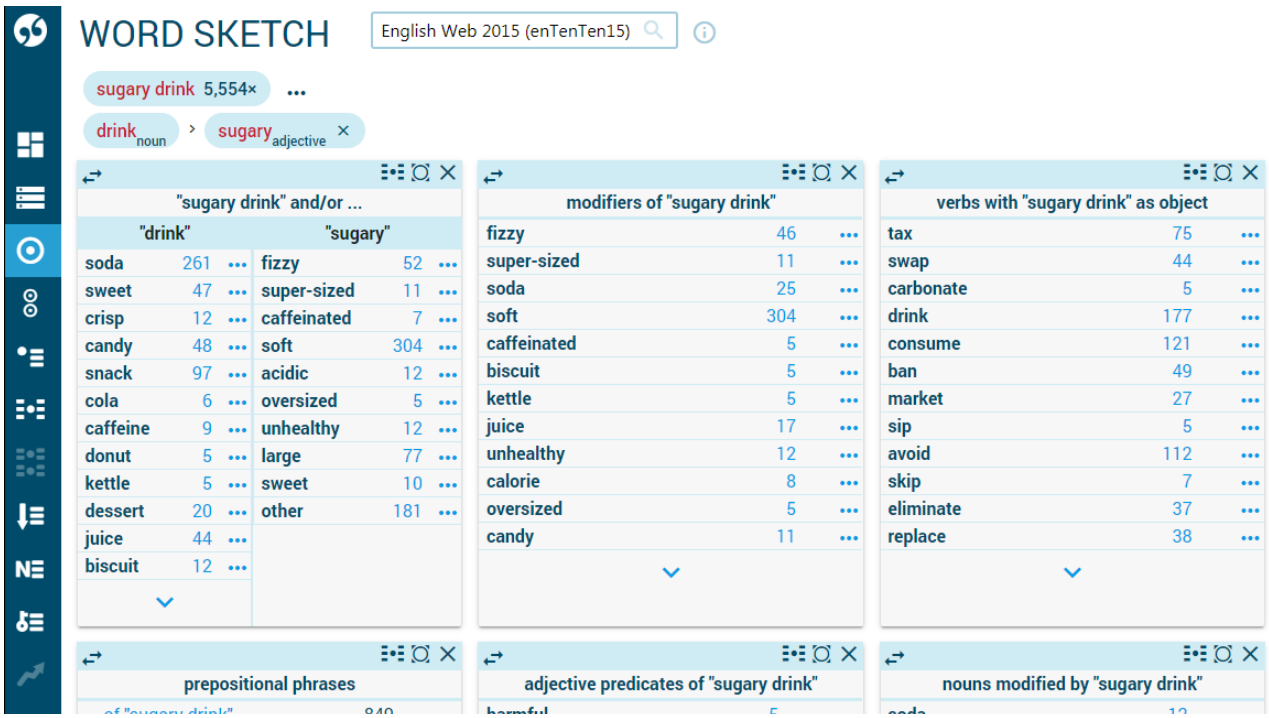


Рисунок 1 – Приклад результатів роботи системи Word Sketch

В корпус менеджері Sketch Engine є ще один інструмент, який вміє працювати з колокаціями. Цей модуль називається «N-grams» та він створює частотні списки колокацій. Користувач має вибір варіантів фільтрації, включаючи регулярні вирази, щоб детально визначити, для яких n-грамів слід генерувати частоти. N-грами можуть бути створені за будь-яким атрибутом, але слово та лема є найбільш часто використовуваними.

Sketch Engine має велику кількість попередньо встановлених корпусів, але користувачі також можуть встановити сторонні корпуси, тобто дану систему можна використовувати з українською мовою. Але для української мови розділ Word Sketch недоступний. [19]

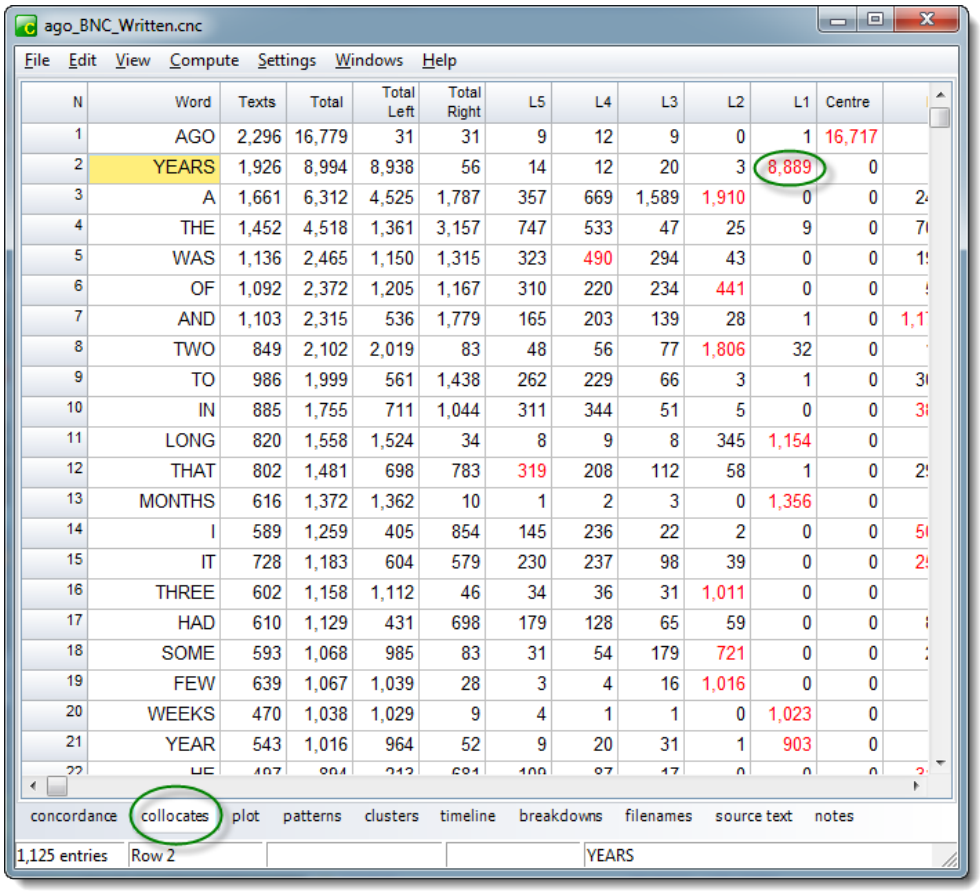
1.3.2. WordSmith Tools 4 (WST 4)

Програмний пакет WST 4 розроблений М. Скоттом та був випущений Oxford University Press. Він забезпечує широкий спектр функцій, що мають відношення до корпусної лінгвістики. Його функції згруповані в три основні категорії: Concord, Keywords та Wordlist. Як випливає з назв категорій, програма може створювати

відповідності, виконувати аналіз ключових слів та складати списки слів за частотою зустрічальності. Іншою помітною особливістю програми є інструмент WebGetter, який дозволяє створювати корпуси на льоту з Інтернету на основі ряду параметрів, включаючи мови, які слід враховувати. Пакет супроводжується вичерпним та інформативним посібником.

На Рисунку 2 можна побачити результати використання WST 4 для знаходження колокацій з двох слів зі словом «ago». Результати, впорядковані за показником логарифмічної ймовірності.

WST 4 здатний вилучати колокації, як загальні, так і на основі ключових слів, а також такі методи для їх пошуку як взаємна інформація, z-критерій та логарифмічна правдоподібність. Недоліками цієї системи можна назвати неінтуїтивний графічний інтерфейс та відсутність підтримки вилучення багатослівних колокацій.



N	Word	Texts	Total	Total Left	Total Right	L5	L4	L3	L2	L1	Centre
1	AGO	2,296	16,779	31	31	9	12	9	0	1	16,717
2	YEARS	1,926	8,994	8,938	56	14	12	20	3	8,889	0
3	A	1,661	6,312	4,525	1,787	357	669	1,589	1,910	0	0
4	THE	1,452	4,518	1,361	3,157	747	533	47	25	9	0
5	WAS	1,136	2,465	1,150	1,315	323	490	294	43	0	0
6	OF	1,092	2,372	1,205	1,167	310	220	234	441	0	0
7	AND	1,103	2,315	536	1,779	165	203	139	28	1	0
8	TWO	849	2,102	2,019	83	48	56	77	1,806	32	0
9	TO	986	1,999	561	1,438	262	229	66	3	1	0
10	IN	885	1,755	711	1,044	311	344	51	5	0	0
11	LONG	820	1,558	1,524	34	8	9	8	345	1,154	0
12	THAT	802	1,481	698	783	319	208	112	58	1	0
13	MONTHS	616	1,372	1,362	10	1	2	3	0	1,356	0
14	I	589	1,259	405	854	145	236	22	2	0	0
15	IT	728	1,183	604	579	230	237	98	39	0	0
16	THREE	602	1,158	1,112	46	34	36	31	1,011	0	0
17	HAD	610	1,129	431	698	179	128	65	59	0	0
18	SOME	593	1,068	985	83	31	54	179	721	0	0
19	FEW	639	1,067	1,039	28	3	4	16	1,016	0	0
20	WEEKS	470	1,038	1,029	9	4	1	1	0	1,023	0
21	YEAR	543	1,016	964	52	9	20	31	1	903	0

Рисунок 2 – Результат пошуку колкацій до слова «ago» методом логарифмічної правдоподібності у WST-4.

1.3.3. Collocate

Ще одна система, яка спеціалізується на виділенні словосполучень з текстів, називається Collocate та була розроблена М. Барлоу. Ця програма дозволяє знаходити колокації, що складаються з декількох слів, максимум з шести слів. Також вона реалізує три методи пошуку словосполучень: t-критерій Стьюдента, взаємна інформація та перевірка відношення правдоподібностей.

Collocate має простий у використанні графічний інтерфейс, в якому всі основні функції доступні з панелі інструментів меню. Програма може працювати з багатьма файлами і має хорошу підтримку XML. Collocate як інструмент, що спеціалізується на видобутку колокації, має дійсно невелику кількість методів виділення колокацій. З іншого боку, даний інструмент простий та зручний у використанні. [21]

1.3.4. Ngram Statistics Package

Даний пакет програмного забезпечення - це спільна робота, головними розробниками якої є Т. Педерсен та С. Банерджі. Цей пакет з відкритим кодом, написаний мовою програмування Perl і може бути описаний як набір програм для аналізу текстових файлів для знаходження n-грамів, і складається він з двох основних програм. Перша програма бере вхідні текстові файли і складає список n-грамів, що містяться у цих файлах. Друга програма бере в якості входу вищезгаданий список n-грамів і запускає метод міри асоціації, обраний користувачем, для того, щоб визначити, які з n-грамів можна розглядати як колокації. Ngram Statistics Package реалізує такі методи міри асоціації як: логарифмічна правдоподібність, взаємна інформація, хі-квадрат, t-критерій та багато інших. Також NSP дуже добре працює з регулярними виразами та може обробляти декілька файлів одночасно, якщо всі вони знаходяться в одному каталозі.

Доступ до функцій NSP здійснюється через інтерфейс командного рядка без графічного інтерфейсу, що може викликати складнощі у користувачів, які не звикли до командного рядка.

Ще одним недоліком даної системи можна назвати формат результатів роботи програми. На Рисунку 3 можна побачити приклад результату роботи програми – файл, який у першому рядку містить число, що вказує на те, скільки в аналізованому тексті було біграм, з наступного рядка перераховуються знайдені результати. На кожній строчці результуючого файлу лексеми розділені знаками ромбів «<>», після останнього токену в строчці є три числа. Перше з цих чисел позначає кількість разів, коли ця колокація зустрічається у вхідному текстовому файлі. Друге та третє число показують скільки разів лексеми зустрічаються на відповідних позиціях в інших знайдених біграмах. Подібний результат виходить і для триграм. [22]

На думку автора, даний формат результуючих даних не є зручним для подальшого використання, адже через розділові знаки, знайдені колокації доведеться додатково очищувати для подальшого використання, що робить використання даного програмного забезпечення, як часини більшої системи дуже незручним.

```
11
line<>of<>2 3 2
of<>text<>2 2 2
second<>line<>1 1 3
line<>and<>1 3 1
and<>a<>1 1 1
a<>third<>1 1 1
first<>line<>1 1 3
third<>line<>1 1 3
text<>second<>1 1 1
```

Рисунок 3 – Приклад роботи NSP для тексту, який містить 11 біграм

У той же час, NSP є дійсно ефективною системою саме для вилучення колокації з текстових даних і була розроблена спеціально з такою метою. [23]

У таблиці 1.1 узагальнено результати аналізу існуючих систем знаходження колокацій у тексті. З чотирьох розглянутих програмних систем три можуть знаходити колокації без ключових слів, що є великою перевагою для даного дослідження. Адже база даних словосполучень буде будуватися на основі всіх знайдених словосполучень у використовуваних текстах. Кожна з розглянутих систем має свої переваги та недоліки. Наприклад, Інструменти WST 4 – це потужний інструмент, який пропонує безліч функцій для дослідження корпусу, але не може знаходити багатослівні колокації. Програмний пакет NSP може це зробити, а також підтримує найбільшу кількість мір асоціації, але дає погано структуровані результати. А система Collocate відповідає більшості критеріям відбору, але підтримує відносно невелику колекцію мір асоціації.

Докладніше розглянемо систему Word Sketch, яка є частиною корпус менеджера Sketch Engine, оскільки вона є найближчим аналогом розроблюваної бази даних словосполучень. Sketch Engine працює з багатьма мовами і текстові корпуси багатьох з них попередньо встановлені і готові для роботи без додаткових зусиль з боку користувача. Українська мова не входить в список попередньо встановлених мов, отже система буде працювати з нею іншим чином. Для української мови доступна функція знаходження всіх n-грамів в корпусі, але нажаль Word Sketch, працювати не буде, адже потрібен корпус с розміткою частин мови та нормалізацією слів, а саме для української мови Sketch Engine не підтримує ці функції. Незважаючи на це Sketch Engine залишається потужним та корисним інструментом для керування текстовим корпусом та дослідження мови.

Отже, розглянуті системи працюють або з готовими розміченими текстовими корпусами, або не працюють з текстовими даними українською мовою. Тільки Sketch Engine N-grams частково відповідає вимогам до розроблюваної системи, але для цього інструменту не можна задати граматичні правила характерні для словосполучень в українській мові, а також ця система використовує один і той самий метод для всіх мов, з якими працює – logDice. Тобто не можна змінити

використовуваний метод та експериментувати з метою визначення найбільш підходящої міри асоціації для конкретної природної мови.

Тому можна зробити висновок, що серед розглянутих систем немає такої, яка повність відповідає вимогам до розроблюваної системи.

Таблиця 1.1 – Результат аналізу існуючих програмних систем

	Sketch Engine	WST 4	Collocate	NSP
Міра асоціації	MI (раніше), logDice коефіцієнт (тепер)	MI, Log-likelihood, z-критерій.	MI, Log-likelihood, t-критерій.	MI, Log-likelihood, хі-квадрат, t-критерій, Фі коефіцієнт, індекс Соренсена, коефіцієнт Жакара та ін.
Вилучення колокацій без ключових слів	+	+	+	+
Вилучення довгих колокацій	+	–	+	+
Підтримка української мови	+ але Word Sketch недоступний	–	–	–

Продовження Таблиці 1.1

	Sketch Engine	WST 4	Collocate	NSP
Графічний інтерфейс користувача	+	+	+	–

1.4. Опис проблем попередньої обробки тексту

Нормалізація тексту – це процес перетворення тексту в єдину канонічну форму, якої раніше він міг не мати. Нормалізація тексту перед його зберіганням або обробкою дозволяє розділити проблеми, оскільки гарантується узгодженість введення перед тим, як над ним виконуватимуться операції. Нормалізація тексту вимагає усвідомлення того, який тип тексту потрібно нормалізувати і як його потім обробляти. Не існує універсальної процедури нормалізації, не існує єдиного правильного підходу чи списку завдань нормалізації, які працюють у всіх ситуаціях. Насправді, проблеми у сфері NLP рідко є типовими, але існує багато цікавих наборів інструментів загального призначення та готових алгоритмів.

Тому список кроків нормалізації, представлений у даному пункті, не є загальними жорсткими правилами, а скоріш рекомендаціями та прийнятими підходами щодо нормалізації тексту. Важливо також зазначити, що в деяких випадках вхідні дані можна не нормалізувати — це випадки, коли більше варіацій і помилок є важливими, наприклад, проблема корекції тесту.

Природна мова має тенденцію дотримуватись природної випадковості її творця. Це означає, що, генеруючи природну мову, ми відображаємо в неї випадкові стани притаманні комбінаціям слів та речень у нашому мовленні. Комп’ютери не так добре справляються з завданням генерації випадковості, хоча це мінімізується за допомогою алгоритмів машинного навчання.

Коли ми нормалізуємо природний мовний ресурс, ми намагаємося зменшити в ньому випадковість, наближаючи його до попередньо визначеного стандарту. Це допомагає зменшити кількість різної інформації, яку має обробляти комп'ютер, а отже, підвищує ефективність.

Нормалізація намагається наблизити дані до нормального розподілу. Це вірно в тому сенсі, що коли ми нормалізуємо природну мову, ми прагнемо зробити так, щоб дані були очікуваними, у зручній та передбачуваній формі, як розподіли ймовірностей, що слідують за нормальним розподілом.

Перш за все, зменшуючи кількість варіацій, ми маємо врахувати менше вхідних змінних для обробки, покращуючи загальну продуктивність і уникаючи псевдо негативних результатів. По-друге, особливо якщо говорити про алгоритми машинного навчання, нормалізація зменшує розмірність вхідних даних, якщо ми використовуємо пласкі старі структури, такі як модель «торба слів» або TF-IDF; або зменшує обсяг обробки, необхідний для створення векторного представлення. По-третє, нормалізація допомагає впоратися з некоректними вхідними даними, що ламають програму, перш ніж вони будуть передані NLP алгоритму прийняття рішень. У цьому випадку ми гарантуємо, що вхідні дані будуть відповідати контракту перед тим, як почнеться їхня обробка.

Нарешті, якщо все зроблено правильно, нормалізація дуже важлива для забезпечення надійного вилучення статистики з вхідних текстових даних.

Виконуючи нормалізацію тексту, треба точно знати, що потрібно нормалізувати і чому. Крім того, призначення та специфіка вхідних даних допомагає сформувати кроки, які будуть застосовані для нормалізації. Є два важливі пункти нормалізації:

- Структура речення – визначити чи завжди речення повинно закінчуватись розділовими знаками, чи можуть бути повторювані знаки пунктуації, або чи варто взагалі видалити всі розділові знаки, крім того, можна використовувати більш конкретну структуру (наприклад, просто витягти підмету та присудок), але її важче досягти.

- Лексика – це одна з основних речей, на які слід звернути увагу, тому що у більшості випадків ми хочемо, щоб наш словниковий запас був якомога меншим, причина в тому, що в алгоритмах NLP слова є ключовими характеристиками, і маючи меншу варіативність, можна досягти кращих результатів.

На практиці можна провести нормалізацію по цих двох аспектах, розбивши їх на простіші задачі. Ось деякі найпоширеніші з них:

- Видалення повторюваних пробілів і пунктуації.
- Видалення наголосу, якщо дані містять діакритичні знаки з іноземних мов – це допомагає зменшити помилки, пов’язані з типом кодування.
- Видалення великої літери – часто робота зі словами з малого регістру дає кращі результати., однак у деяких випадках великі літери дуже важливі для отримання інформації, як-от імена та назви місць.
- Видалення або заміна спеціальних символів та емодзі, наприклад, видалення хештегів.
- Нормалізація формату дат, номери телефонів або інших даних, які мають стандартний формат.
- Виправлення орфографії— це дуже важливо, якщо вхідні дані надходять напряму від кінцевих користувачів, наприклад, пости в соціальних мережах, миттєві повідомлення та електронні листи.
- Видалення варіацій роду або часу та приведення однокореневих слів до одного виду за допомогою стеммінгу або лемматизації.
- Заміна рідкісних слів на більш поширені синоніми.
- Видалення стоп-слів.

Слова в словосполученнях зустрічаються в реченнях в різних формах, в результаті чого утворюються варіації одного словосполучення. Щоб об’єднати ці форми в одну, кожне слово має бути лематизоване, тобто потрібно знайти лему для даної форми. Контекст слова не враховується, що інколи призводить до неоднозначної лематизації.

Іншою лінгвістичною інформацією, отриманою шляхом лематизації, є частини мови. Також додаткової уваги заслуговують стоп-слова. Стоп-слова – це слова, які дуже часто зустрічаються в письмовій або усній комунікації природною мовою, тому їх іноді розглядають як шум сигналу в каналі, якщо розглядати через модель інформації Шеннона. [24] У багатьох програмах, які аналізують текстові дані стоп-слова спочатку відфільтровуються, перш ніж виконувати будь-яку іншу обробку тексту. Стоп-слова зазвичай включають прийменники, сполучники, числівники та займенники.

Отже, для того щоби вибрати, які саме кроки нормалізації текстових даних обрати, необхідно проаналізувати конкретне завдання. Розроблення алгоритму попереднього оброблення текстових даних розглянуте в пункті 2.1.

1.5. Постановка завдання дослідження

Інтернет зробив вивчення та дослідження практично будь-якої живої мови легшим, ніж будь-коли. Мова вивчається і досліджується не лише для використання у філології, а й для поліпшення машинного перекладу та інструментів обробки природної мови, для пошукових систем і голосових помічників, а також інструментів для редагування та узагальнення текстів та алгоритмів автоматичного генерування природної мови. Щоб бути ефективними, алгоритми, які працюють із живою мовою, повинні постійно аналізувати справжнє мовлення, оскільки мови постійно змінюються та трансформуються. Джерелом даних для досліджень у сфері NLP можуть стати різноманітні ресурси – такі як словники, перекладачі та майже будь-який текст в Інтернеті.

Метою даного дослідження є розроблення бази даних словосполучень, що генерується шляхом аналізу великої кількості текстових даних та знаходження в них всіх можливих варіацій сполучень слів. Така база даних саме може служити не тільки як словник, але й використовуватися як ресурс для дослідження мови за допомогою комп'ютерів.

За визначеною метою дослідження можна визначити ряд вимог до розроблюваної бази даних словосполучень української мови:

1. Розроблювана база даних повинна наповнюватися в напівавтоматичному режимі, тобто такий словник не буде наповнюватися людьми, а все що потрібно, аби додати нові дані в систему – це подати на вхід джерело текстових даних.
2. Вхідні дані повинні бути нормалізовані системою.
3. Система повинна автоматично знаходити всі можливі варіанти словосполучень.
4. Знайдені словосполучення повинні бути відфільтровані та збережені у сховищі даних.
5. Система повинна підтримувати пошук по отриманим в результаті аналізу даним.
6. Розроблювана система має працювати ефективно, надійно та враховувати можливість аналізу великих даних і робити це достатньо швидко.
7. Система має знаходити приклади використання шуканих користувачем словосполучень, може використовувати для цього сторонні словники та інші мовні ресурси або приклади з вхідних текстових даних.
8. Розроблювана система повинна мати зручне API для виконання запитів пошуку та отримання даних.

Висновки до першого розділу

В цьому розділі було сформульовано проблему автоматичного виділення словосполучень з текстових даних та дано визначення поняттю словосполучення.

Проаналізувавши основні методи знаходження колокацій у текстах можна зробити висновок, що не існує одного загальноприйнятого підходу, який б працював для будь-якого тексту будь-якою мовою. Всі методи мають певні переваги та недоліки і відрізняються один від одного. На цю тему проведена велика кількість досліджень, але переважна більшість з них були направлені на дослідження текстів англійською мовою. Тобто особливості інших мов, а саме української мови враховані не були. Але всі ці дослідження показують варіативність підходів та важливість даного розділу NLP, а також те, що ця проблема є актуальною та викликає науковий інтерес.

Були також проаналізовані існуючі подібні рішення та визначено, що серед розглянутих немає такої системи, яка б повністю відповідала поставленому завданню. Проаналізовані системи допомогли сформулювати основні вимоги до розроблюваної бази даних словосполучень української мови.

Розглянуто також досить нетривіальну проблему нормалізації тексту і визначено, що набір кроків нормалізації є специфічним для кожного завдання. Алгоритм нормалізації, що буде використовуватися в розроблюваному програмному забезпечення буде розглянутий у наступному розділі.

Було також поставлено завдання на дослідження, сформовано мету та основні вимоги до розроблюваної бази даних словосполучень української мови.

2. МЕТОДИ АВТОМАТИЧНОГО ЗНАХОДЖЕННЯ ТА ФІЛЬТРАЦІЇ СЛОВОСПОЛУЧЕНЬ

2.1. Вибір алгоритму попередньої обробки тексту

Пошук колокацій за допомогою статистичних методів насправді є їх ранжуванням щодо міри асоціації. Ці показники базуються на частотах слів та послідовностях слів у корпусі, які отримані шляхом попередньої обробки корпусу. В даному випадку попередня обробка корпусу означає токенізацію, лемматизацію та розмітку за частинами мови слів у корпусі, а також підрахунок того, скільки разів кожне слово та n-грама з’являються в корпусі. У цьому розділі буде формалізована попередня обробка текстових даних для подальшого використання мір асоціації на цих даних з ціллю знаходження в них всіх можливих колокацій.

Як було визначено у попередньому розділі – кроки нормалізації обираються під конкретне завдання. Отже, для задачі автоматичного пошуку словосполучень за допомогою статистичних методів були визначені наступні кроки:

1. видалення повторюваних пробілів, пунктуації, посилань та номерів,
2. виправлення орфографії,
3. стемінг та лематизація,
4. видалення стоп-слів.

Дуже важливою ознакою нормалізації є те, що порядок кроків має значення. Якщо не бути обережним, можна видалити інформацію, важливу для майбутніх кроків, наприклад, видалити стоп-слова перед лематизацією. Можна навіть розділити ці кроки на дві послідовні групи: «кроки до токенізації» – для кроків, які змінюють структуру речення, і «кроки після токенізації» – для кроків, які змінюють лише окремі токени, щоб уникнути дублювання кроків.

Розглянемо кожен крок докладніше та проаналізуємо результати їх застосування за допомогою статистики.

2.1.1 Видалення повторюваних пробілів, знаків пунктуації, номерів та URL-адрес

Даний крок нормалізації може бути виконаний за допомогою простої заміни за допомогою регулярного виразу. Треба скласти регулярні вирази, які включають в себе всі номери та знаки пунктуації та зробити заміну знайдених таким виразом символів на пусту строку. Таким чином всі знайдені непотрібні для аналізу символи будуть видалені з текстових даних.

Також потрібно видалити URL-адреси, оскільки це значно зменшує кількість наявних унікальних токенів. Видалення URL-адрес повинно передувати видаленню пунктуації, оскільки, заміна всіх символів пунктуації може зламати структуру посилання і потім його неможливо буде знайти.

Посилання видаляються за допомогою регулярних виразів, але скласти регулярний вираз на всі види посилань не є тривіальним завданням, тому може бути обраний вираз, який допоможе знайти якомога більше варіантів посилань.

2.1.2 Виправлення орфографії

Цей крок нормалізації може викликати деякі небажані зміни через те, що у більшості словників для виправлення орфографії відсутні слова, що є важливими для формування контексту, тому можуть бути хибно-позитивні результати – алгоритм знайде орфографічні помилки там, де їх немає. Тому використовувати цей крок нормалізації потрібно свідомо. Є різні способи виправляти орфографію автоматично. В даному дослідженні використаний модуль на ім’я Hunspell, який є дуже швидким, працює з українською мовою та є досить ефективним. Інший спосіб зробити це – навчити модель глибокого навчання виправляти орфографію на основі контексту, що буде більш ефективно, але це вже окрема важлива задача NLP.

Hunspell — це безкоштовна бібліотека перевірки орфографії, морфологічний аналізатор та інструмент командного рядка.

Hunspell використовується офісним пакетом LibreOffice, безкоштовними браузерами, такими як Mozilla Firefox і Google Chrome, а також іншими інструментами та ОС, такими як дистрибутиви Linux і macOS. Це також інструмент командного рядка для Linux, Unix-подібних та інших операційних систем.

Він розроблений для швидкої та якісної перевірки та виправлення орфографії для мов із системою письма на рівні слів, включаючи мови з багатою морфологією, складним складанням слів та кодуванням символів.

Основні особливості бібліотеки Hunspell:

- Дуже гнучкі налаштування пропозицій: таблиці заміни частин слів, фонетичні та інші альтернативні транскрипції на рівні основи потрібні, щоб розпізнавати та виправляти всі типові орфографічні помилки, не пропонувати ненормативну лексику тощо.
- Складна морфологія: словник і омонімія афіксів; подвійне видалення афіксів для обробки флексійних і деривaційних груп морфем для аглютинативних мов, таких як азербайджанська, баскська, естонська, фінська, угорська, турецька; умовні афікси, нульові морфеми, заборонені слова тощо.
- Робота зі складними сполуками: розпізнавання сполук, що складаються з довільної кількості слів, обробка афіксів у складі тощо.
- Спеціальні словники з афіксацією.
- Стемінг.
- Морфологічний аналіз.
- Морфологічна генерація
- Підтримка Unicode.

Система Hunspell була розроблена для угорської мови, яка є дуже складною в плані морфології. Ця система також добре працює з українською мовою, саме тому вона була обрана для нормалізації для виправлення орфографії. На Рисунку 4 можна побачити, що Hunspell визначає чи вірний правопис слова та підказує можливі варіанти виправлення. [25]

```
[10] import hunspell
     spellchecker = hunspell.HunSpell(
         '/usr/share/hunspell/uk_UA.dic',
         '/usr/share/hunspell/uk_UA.aff',
     )
```

```
[11] spellchecker.spell('орфографія')

     True
```

```
[12] spellchecker.spell('офографія')

     False
```

```
[13] spellchecker.suggest('офографія')

     ['орфографія',
      'ороrрафія',
      'фонографія',
      'фотографія',
      'фізіографія',
      'зоографія']
```

Рисунок 4 – Виправлення правопису для слова «орфографія» за допмогою Hunspell

2.1.3 Стемінг та лематизація

У лінгвістичній морфології та інформаційному пошуку стемінг — це процес зведення флексійних та іноді похідних слів до їх основи чи кореня. Основа не обов'язково має бути ідентичною морфологічному кореню слова; зазвичай достатньо, щоб споріднені слова відображалися в одній основі, навіть якщо ця основа сама по собі не є дійсним коренем. Багато пошукових систем розглядають слова з однаковою основою як синоніми для своєрідного розширення запиту – процес, який називається злиттям.

Даний крок нормалізації є корисним для задачі пошуку словосполучень за допомогою статистичних методів тому, що зменшується кількість унікальних слів. Таким чином можна буде знайти більше колокацій, через те, що слова у різній

формі, але з однією основою, після нормалізації стануть одним токеном у множині всіх слів, що залишаться на подальший аналіз.

Для стемінгу в даному дослідженні також використаний Hunspell. Метод його API Stem дозволяє отримати основу слова, працює швидко та досить точно для української мови. Функція Hunspell Stem шукає слова зі словника, які відповідають кореню даного слова, функція повертає список, оскільки деякі слова можуть мати кілька збігів. На Рисунку 5 можна побачити результати роботи Hunspell для стемінгу слова «працює».

```
[28] stem = spellchecker.stem('працює')
     stems = [s.decode(spellchecker.get_dic_encoding()) for s in stem]

[29] stems

     ['працювати']
```

Рисунок 5 – Стемінг слова «працює» за допомогою Hunspell

2.1.4 Видалення стоп-слів

Стоп-слова або шумові слова— це слова, які не несуть важливого значення і зазвичай вилучаються з текстів при їх обробці комп'ютерами. До загальних стоп-слів належать окремо розташовані букви алфавіту, займенники, прийменники, вигуки та ін. У відкритому доступі знаходяться списки стоп-слів української мови, тобто їх можна видалити зі списку слів отриманих після попередніх кроків нормалізації простим пошуком.

Отже, для вирішення задачі пошуку словосполучень в тестових даних за допомогою статистичних методів, були обрані кроки нормалізації та визначений порядок їхнього виконання.

2.1.5 Результати нормалізації вхідних текстових даних

Тепер визначимо які результати застосування цих кроків нормалізації. Для визначення деяких показників був проведений експеримент – описані вище кроки нормалізації були виконані на текстах корпусу UA-GEC та підрахована статистика, яка буде описана нижче. [26]

Нормалізація тексту найкраще відіграє свою роль у застосуванні до алгоритмів NLP, підвищуючи ефективність, точність та інші відповідні показники. В даному підрозділі буде вказано на деякі переваги, які можна чітко побачити за статистикою.

По-перше, загальна кількість окремих слів значно зменшилася. У цьому конкретному випадку було зменшено кількість слів приблизно на 57%, що можна побачити на Рисунку 6.

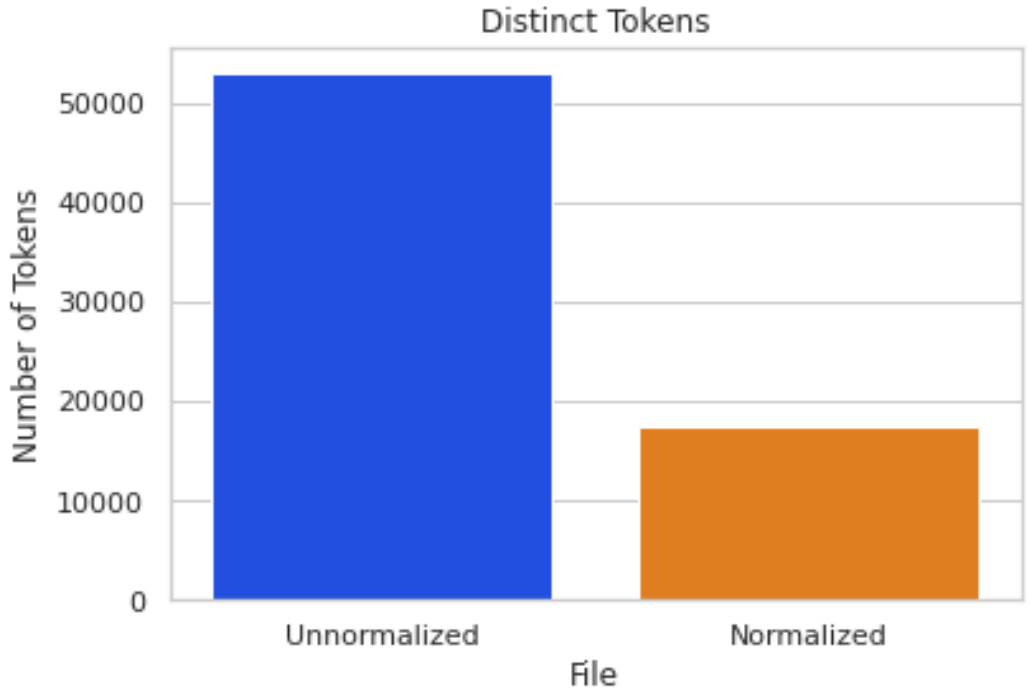


Рисунок 6 – Зменшення кількості унікальних слів після нормалізації.

Більша різниця спостерігається в кількості слів, що часто зустрічаються. Ці слова відповідають приблизно 80% усіх слів, зазвичай приблизно 10–20% лексем становлять 80% тексту.

На Рисунку 7 можна побачити, що застосувавши нормалізацію, кількість найпоширеніших слів було зменшено на 82%, що дуже багато. Це також означає, що будь-яка техніка машинного навчання або алгоритм NLP, з якою будуть використані ці дані, зможе ефективніше узагальнювати та давати кращі результати.

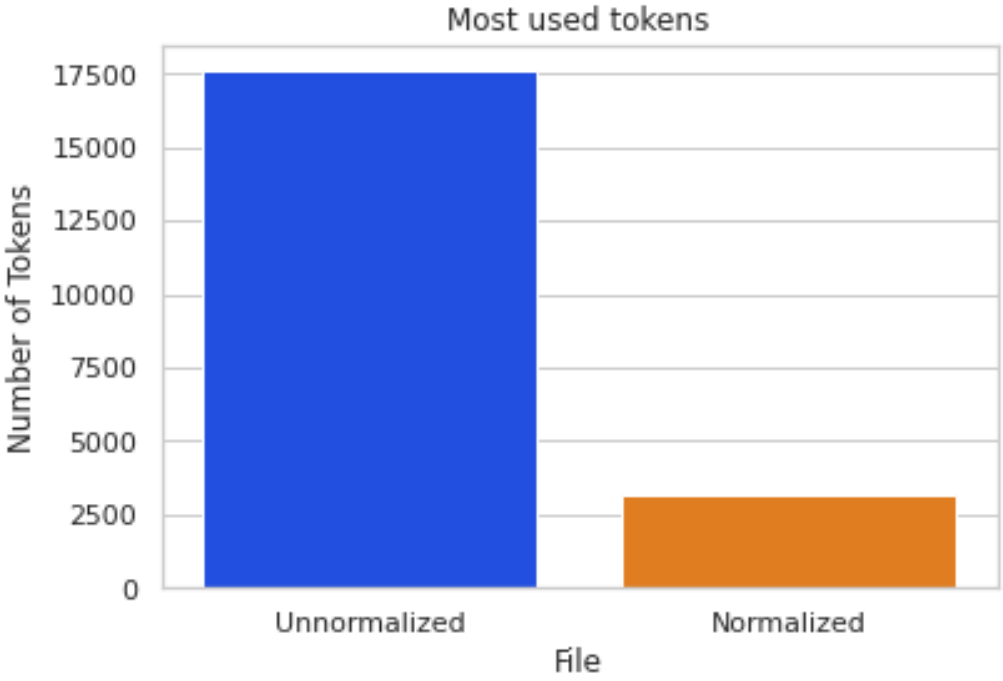


Рисунок 7 – Зменшення кількості найпоширеніших слів після нормалізації.

Ще одна важлива річ, яка стосується нормалізації тексту, полягає в тому, що для того, щоб вона була корисною, нормалізований текст повинен зберегти структуру природної мови. Ми можемо побачити це за самим розподілом даних. Одним із показників є те, що, якщо все зроблено правильно, речення після нормалізації не будуть набагато меншими чи більшими. На гістограмі на Рисунку 8 можна побачити, що довжина речень після нормалізації, майже повністю відповідає структурі ненормованих даних. А на Рисунку 9 видно, що стало більше унікальних слів у реченнях, це каже про те, що проведення нормалізації є ефективним і допомагає прибрати «шум» з текстових даних, таким чином

зменшується кількість повторюваних слів. Також можна побачити, що крива має тенденцію бути близькою до кривої нормального розподілу.

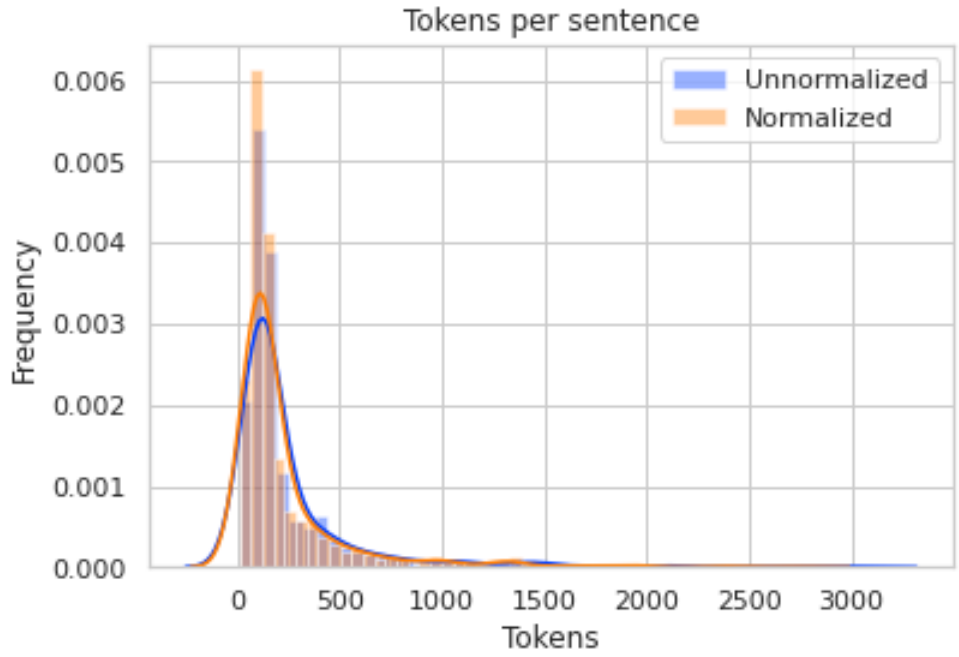


Рисунок 8 – Розподіл кількісті слів у реченнях до та після нормалізації.

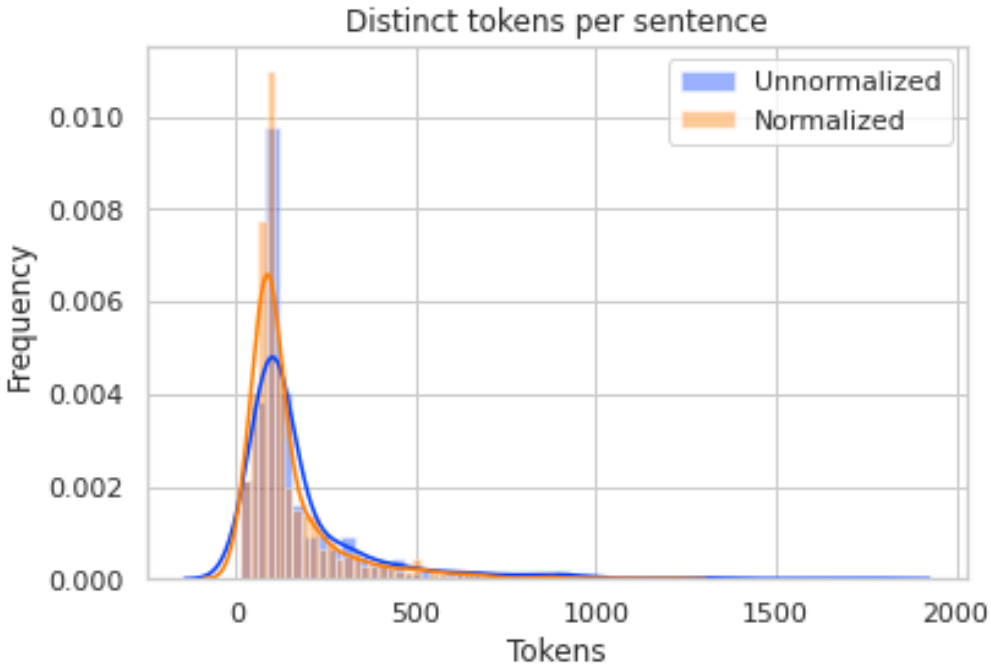


Рисунок 9 – Розподіл за кількістю унікальних слів у реченнях до та після нормалізації.

Наступні діаграми розмаху, що можна побачити на Рисунку 10 та Рисунку 11 показують, як розподіляються дані, включаючи медіану, квартилі, мінімальне та максимальне значення. В ідеалі медіана повинна бути такою ж, або дуже близькою, до ненормованих даних. Бажано також, щоб стандартне відхилення у розподілі нормованих даних було наближене до ненормованих. Якщо стандартне відхилення збільшується, це означає, що навколо медіани ми маємо більше даних, ніж до нормалізації і це добре. Крім того, можна побачити, що зменшилася кількість даних, які не включені між вусами.

Видно, що після нормалізації збільшився міжквартильний діапазон, в якому знаходиться більшість найбільш використовуваних слів. Також було збережено медіану та зменшилася кількість даних поза вусами. Це означає, що текстові дані не були зіпсовано, але були змінені та стали менш складними, ніж були до нормалізації.

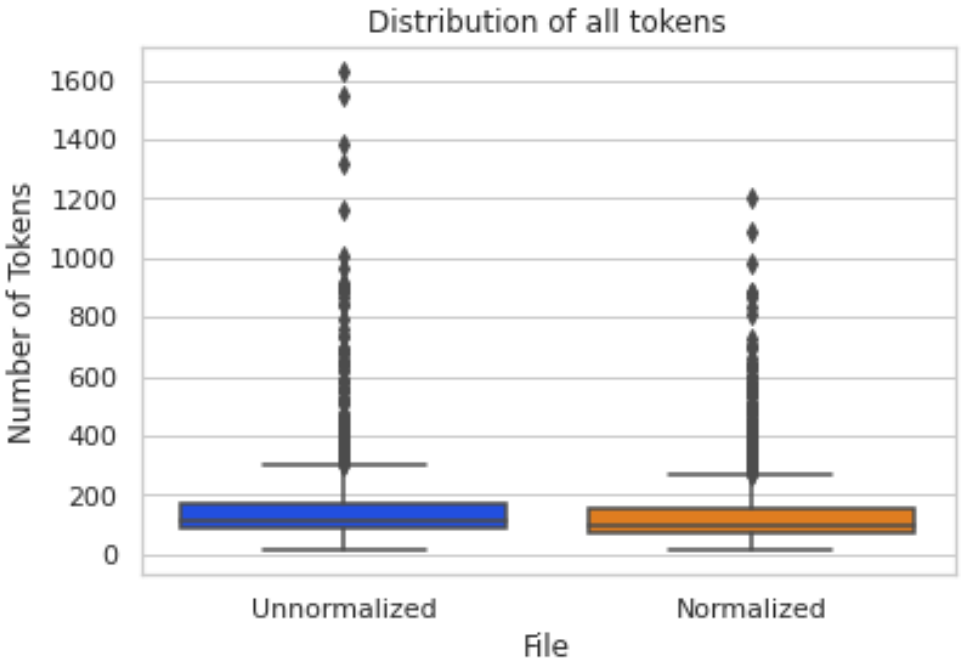


Рисунок 10 – Діаграма розмаху розподілу всіх слів до та після нормалізації.

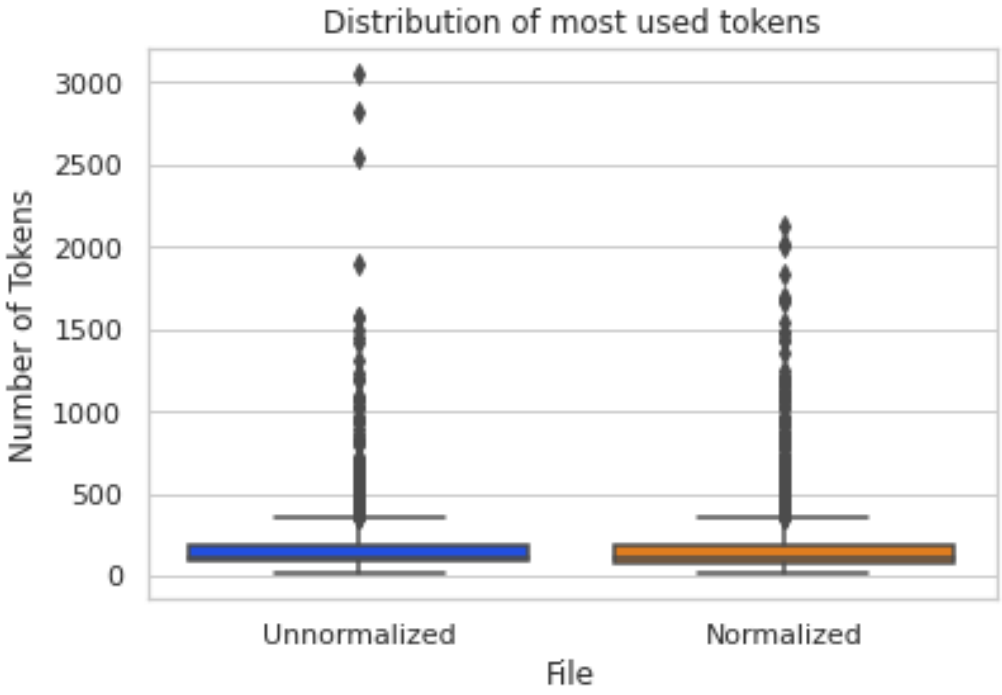


Рисунок 11 – Діаграма розмаху розподілу найбільш використовуваних слів до та після нормалізації.

Отже, в даному пункті було проаналізовано для чого потрібна нормалізація текстових даних та які кроки обрані для вирішення завдання автоматичного пошуку словосполучень в текстах українською мовою. Крім того, був проведений експеримент, щоб довести, що нормалізація є ефективною та дійсно вирішує поставлені перед нею завдання. Дійсно видно, що загальна кількість слів скоротилася, зникла пунктуація та найбільш поширені слова залишилися маже такі як до нормалізації, навіть вдалося трохи розширити область стандартного відхилення для розподілу унікальних слів у реченнях після нормалізації, що означає – текст став простіший для аналізу алгоритмами NLP, але не втратив важливу інформацію, яка в ньому міститься.

2.2. Вибір методу знаходження словосполучень

В даному дослідженні пошук словосполучень буде відбуватися засобами статистичного аналізу текстових даних. Методи, які використовуються для статистичного аналізу називаються – міри асоціації.

Міри асоціації використовуються для визначення сили асоціації двох слів. Треба зауважити, що мова йде про «два слова», оскільки всі міри асоціації спочатку було визначено для діаграм, тому всі існуючі міри для n-грам, де n > 2, в основному є запропонованими розширенням методів для біграм. Вибір відповідної міри асоціації має вирішальне значення для всього процесу пошуку словосполучень, оскільки ці методи використовуються, щоб визначити, чи є n-грам словосполученням.

Міри асоціації можна приблизно розділити на чотири категорії:

- Сортування простим підрахунком — це найпростіша міра, де кожен n-грам отримує оцінку, рівну його частоті зустрічальності в корпусі.
- Методи перевірки відношенням правдоподібностей — це методи, що перевіряють нульову гіпотезу, яка стверджує, що між словами немає зв’язку за межами окремих випадків, вони працюють шляхом обчислення ймовірності p того, що подія відбудеться, якби H_0 істинно, а потім відхиляють H_0, якщо p занадто низьке, використовуючи певний рівень значущості.
- Інформаційно-теоретичні міри — типовим представником цього класу є міра взаємної інформації, яка говорить нам, наскільки збільшується інформація, яку ми маємо про знаходження одного слова в позиції $i + 1$, якщо нам надати інформацію про зустріч іншого слова в позиції i.
- Евристичні методи — різні автори намагалися визначити власні міри визначення колокації, жодна з них не має сильного формального обґрунтування, але всі вони висловлюють ідею, що два слова, швидше за все, будуть словосполученням, чим більше вони з’являються разом, і тим менше вони з’являються одне без одного.

Прикладами евристичних методів є коефіцієнт Кульчинського, коефіцієнт Фагера і Макгоуена, коефіцієнт Дайса, коефіцієнт Юла тощо.

Є також кілька способів знаходження словосполучень, що використовують більше, ніж інформацію про міри асоціації та частини мови. Пірс у своєму дослідженні використовує той факт, що колокації не є композиційними, тому він використовує інформацію про синоніми з WordNet, замінюючи одне з слів його синонімом, щоб побачити, чи відповідає біграма-кандидат цій властивості. Він підраховує, скільки разів ця нова біграма, утворена з синонімом, зустрічається в корпусі, і якщо немає суттєвої різниці між частотою двох варіантів, то діграма не може бути колокацією, оскільки вона, очевидно, є композиційною. [17]

Дуже великий список з 84 мір асоціації можна знайти в роботі П. Пеціна. В цьому дослідженні також наведений інший цікавий спосіб виділення колокацій, де автор намагається поєднати результати роботи кількох мір асоціації, щоб визначити, чи є n-грам словосполученням. Значення отримані в результаті роботи різних мір асоціації розглядаються як ознаки кожної n-грами, і разом з набором вручну знайдених колокацій і неколокацій, завдання знаходження колокацій стає завданням класифікації за допомогою деякого алгоритму машинного навчання. [27]

Порівнюючи міри асоціації, ми спочатку повинні вирішити, як та над якими мірами провести тест. Наприклад, Еверт і Кренн [7] порівнювали t-критерій, простий підрахунок, коефіцієнт логарифмічної правдоподібності та хі-квадрат, тоді як Танопулос [16] порівняли t-критерій, взаємну інформацію, хі-квадрат і логарифмічну правдоподібность.

В даному дослідженні були порівнянні такі показники: простий підрахунок, поточкова взаємна інформація, логарифмічна правдоподібність, хі-квадрат і t-критерій.

Далі будуть описані міри асоціації, які у будуть порівняні між собою та використані для генерації розроблюваного словника словосполучень.

2.2.1. Метод простого підрахунку частот

Метод підрахунку частот є найпростішим і мабуть найпершим підходом використаним для знаходження колокацій. У цій техніці частота біграм або частота слів, які одночасно зустрічаються в певному вікні, вказує на те, чи є сполучення слів колокацією чи ні. Комбінації впорядковуються за частотою для створення списку кандидатів, а комбінації, що найбільш часто зустрічаються, приймаються як колокації. Хоча деякі з кандидатів, які опиняються у верхній частині списку, є колокаціями, інші є парами функціональних слів. Щоб відкинути кандидатів, які не підпадають під визначення словосполучення, у багатьох існуючих дослідженнях рекомендується фільтрація, наприклад, за частинами мови.

Передбачувано, що лише вибір біграм, які найчастіше зустрічаються, не є дуже цікавим з точки зору лінгвістики. Однак існує дуже проста евристика, яка значно покращує ці результати Дж. Джастесон і С. М. Кац пропонують пропускати фрази-кандидати через фільтр за частинами мови, який пропускає лише ті кандидати, які відповідають певному шаблону. Вони запропонували шаблони, які можна побачити в таблиці 2.1. В даних шаблонах A означає прикметник, P — прийменник, а N — іменник. [28]

Таблиця 2.1 – Шаблони тегів частини мови для фільтрації спільного розташування

Шаблон тегів частин мови	Приклад
AN	нормальний розподіл
NN	коефіцієнт регресії
AAN	стандартний нормальний розподіл
ANN	статистична обробка даних
NAN	закон великих чисел
NNN	функція розподілу ймовірностей

Продовження Таблиці 2.1

Шаблон тегів частин мови	Приклад
NPN	програма без інтерфейсу

Метод Джастесон і Кац для виявлення колокацій є дуже цікавим, оскільки він демонструє важливий момент. Простий підрахунок частот у поєднанні з невеликою кількістю лінгвістичних знань, в даному випадку враховано важливість частин мови, може дати дуже добрі результати.

Більш детально фільтрація кандидатів знайдених статистичними методами буде розглянута у пункті 2.3 даного розділу.

2.2.2. Перевірка гіпотези незалежності

Для того щоб вирішити, чи є знайдена пара слів словосполученням, необхідно довести, що спільна зустріч даних слів — це більше, ніж збіг. Поширеним підходом, що показує залежність між словами, є перевірка гіпотези незалежності. Методи перевірки гіпотез намагаються відхилити нульову гіпотезу, яка стверджує, що слова в комбінації незалежні одне від одного.

Дійсно важливо знати, чи зустрічаються два слова разом частіше, ніж випадково. Оцінка того, чи є щось випадковою подією, є однією з класичних проблем статистики і зазвичай це формулюється в термінах перевірки гіпотез. Треба сформулювати нульову гіпотезу H_0 про те, що між словами немає зв'язку, крім окремих випадків, обчислюємо ймовірність p того, що подія станеться, якщо H_0 є істинною, а потім відхиляємо H_0, якщо ймовірність p занадто низька та залишаємо H_0 в іншому випадку.

Важливо зазначити, що даний спосіб аналізу даних, передбачає розгляд двох речей одночасно. Як і в інших методах, ціль – відшукати певні закономірності в даних, але також враховується, кількість проаналізованих даних. Навіть якщо є явна закономірність, вона не буде врахована, якщо не буде проаналізовано достатньо даних, щоб впевнитися, що ця закономірність не є просто випадковістю.

Методологія перевірки гіпотез широко застосовується до проблеми автоматичного пошуку словосполучень в текстових даних. Спочатку потрібно сформулювати нульову гіпотезу, яка має бути істинною, якщо два слова не утворюють колокацію. Для такої вільної комбінації двох слів ми будемо вважати, що кожне зі слів w^1 і w^2 генерується повністю незалежно від іншого, і тому їхні шанси зібратися просто дають:

$$P(w^1 w^2) = P(w^1)P(w^2) \quad (1)$$

Модель передбачає, що ймовірність того, що вони зустрінуться разом, є лише добутком ймовірностей зустріти окремі слова. Ця модель є досить спрощеною і емпірично не точною, але наразі незалежність приймається за нульову гіпотезу.

2.2.3. t-критерій Стьюдента

t-критерій Стьюдента — це статистичний тест перевірки гіпотези, в якому в якості статистичного критерію використовується t-розподіл Стьюдента за нульовою гіпотезою.

Цей метод найчастіше застосовується, коли тестова статистика слідує нормальному розподілу, якщо відоме значення коефіцієнту масштабу в тестовій статистиці. Коли коефіцієнт масштабу невідомий і замінений оцінкою на основі даних, статистика тесту за певних умов відповідає t-розподілу Стьюдента. Наприклад, t-критерій можна використовувати, щоб визначити, чи сильно відрізняються один від одного середні значення двох наборів даних.

Критерій Стьюдента широко використовується для пошуку колокацій. У t-критерії нульова гіпотеза стверджує, що вибірка взята з нормального розподілу із середнім значенням μ. Тест розглядає відмінності між очікуваними та спостережуваними середніми, масштабованими за дисперсією даних. В результаті,

якщо спостережуване середнє відрізняється від очікуваного, нульова гіпотеза відхиляється.

При перевірці нульової гіпотези про те, що середня сукупність дорівнює заданому значенню μ, використовується t статистика, яка обчислюється за формулою (2).

$$t = \frac{\bar{x} - \mu}{\sqrt{\frac{s^2}{N}}} \tag{2}$$

де $\bar{x}$ – вибіркове спостережуване середнє, s^2 – стандартне відхилення вибірки, а N – розмір вибірки, а μ – середнє значення розподілу.

Якщо t статистика достатньо велика, нульова гіпотеза може бути відхилена. Щоб застосувати t-критерій незалежності двох слів w^1 і w^2, припускаючи, що $f(w^1), f(w^2)$, і $f(w^1 w^2)$, є відповідними частотами w^1, w^2, і $w^1 w^2$, а N — загальна кількість слів у корпусі , можуть бути дані наступні ймовірності

$$P(w^1) = \frac{f(w^1)}{N} \tag{3}$$

$$P(w^2) = \frac{f(w^2)}{N} \tag{4}$$

$$P(w^1 w^2) = \frac{f(w^1 w^2)}{N} \tag{5}$$

В t-критерії нульовою гіпотезою є

$$P(w^1 w^2) = P(w^1)P(w^2), \tag{6}$$

і якщо нульова гіпотеза вірна, то буде присвоєне значення 1 для результату w^1w^2 і 0 для будь-якого іншого результату, що відповідає розподілу Бернуллі із середнім значенням.

$$\mu = P(w^1)P(w^2) \tag{7}$$

За використання біноміального розподілу, вибіркове середнє значення дорівнює

$$\bar{x} = P(w^1w^2), \tag{8}$$

а дисперсія вибірки дорівнює

$$s^2 = P(w^1w^2)(1 - P(w^1w^2)) \approx P(w^1w^2), \tag{9}$$

оскільки $P(w^1w^2)$ є малим для більшості n-грам.

Значення t для всіх n-грам обчислюються та порівнюються із значенням t на попередньо визначеному рівні значущості. Нульова гіпотеза відхиляється для комбінацій, які мають вищі значення t, ніж значення в таблиці t.

2.2.4. Критерій Хі-квадрат Пірсона

Використання t-критерію для пошуку колокації було піддано критиці в роботі Черча та Мерцер, оскільки він припускає, що ймовірності розподілені приблизно нормально, що загалом не відповідає дійсності. [29] Альтернативним

тестом на залежність, який не передбачає нормального розподілу ймовірностей, є критерій хі-квадрат.

Отже, критерій хі-квадрат — це методика перевірки гіпотези, представлена Пірсоном, яка не вимагає нормально розподілених ймовірностей, як у t-критерії. Тест застосовується до таблиць 2x2, щоб порівняти спостережувані частоти з очікуваними частотами, та перевірити, чи можна відхилити нульову гіпотезу незалежності. Розраховується очікувана частота, що представляє незалежність слів, і якщо спостережувані частоти відрізняються від очікуваної частоти, нульова гіпотеза відкидається.

Статистика хі-квадрат підсумовує відмінності між спостережуваними O_{ij} та очікуваними значеннями E_{ij} у всіх клітинках таблиці та масштабує відмінності за величиною очікуваного наступним чином:

$$x^2 = \sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{10}$$

де i — рядок, а j — індекс стовпця в таблиці. Очікувана частота кожної клітинки обчислюється на основі підсумків рядків і стовпців, перетворених у пропорції. [30]

Значення x^2 обчислюється для всіх n-грам у корпусі та формується рейтинговий список. Комбінації, що мають вищі значення, приймаються як колокації.

Загалом, для задачі пошуку колокацій відмінності між статистикою t і статистикою хі-квадрат не дуже великі. Наприклад, двадцять біграм з найвищими значеннями за t статистикою будуть тими самими найкращими двадцятьма біграмами за статистикою хі-квадрат. Однак критерій хі-квадрат також підходить для великих ймовірностей, для яких припущення нормальності t-критерію є недійсним. Можливо, причиною цього є те, що критерій хі-квадрат був застосований до більш широкого кола проблем у пошуку спільного розташування слів.

Подібно до того, як застосування t-критерію є проблематичним через основне припущення нормальності розподілу статистики, так само застосування хі-квадрат у випадках, коли числа в таблиці 2x2 малі.

2.2.5. Логарифмічна правдоподібність

Ще один підхід до перевірки гіпотез – це перевірка відношенням правдоподібностей імовірності. Даний метод більше підходить для розріджених даних, ніж критерій хі-квадрат, але статистика методу відношення правдоподібностей, є легшою для інтерпретації, ніж статистика хі-квадрат. Адже, це просто число, яке говорить нам, наскільки одна гіпотеза є вірогіднішою за іншу.

Застосовуючи тест відношення правдоподібностей, представлений Т. Данінгом, до пошуку колокацій, ми досліджуємо наступні два альтернативні пояснення частоти зустрічальності біграми w^1w^2

$$H_1: P(w^1|w^2) = p = P(w^2|\neg w^1) \tag{11}$$

$$H_2: P(w^2|w^1) = p_1 \neq p_2 = P(w^2|\neg w^1) \tag{12}$$

Перша гіпотеза H_1 стверджує, що поява w^2 не залежить від попереднього входження w^1. Тоді як, друга гіпотеза H_2 навпаки формалізує залежність словосполучення. Якщо H_1 прийнята, комбінація не є колокацією, а якщо прийнята H_2, комбінація розглядається як колокація. Оцінки максимальної правдоподібності для p, p_1 та p_2 наведено нижче на формулах (13, 14, 15) відповідно.

$$p = \frac{c}{N} \tag{13}$$

$$p_1 = \frac{c_{12}}{c_1} \tag{14}$$

$$p_2 = \frac{c_2 - c_{12}}{N - c_1} \tag{15}$$

де c_1, c_2, c_{12} – кількість входжень w^1, w^2 та w^1w^2 відповідно; N — загальна кількість слів у корпусі.

Припускаючи, що біноміальний розподіл ($b(k;\ n,x)\ =\ xk(1\ -\ x)n\ -\ k$) відношення логарифмічної правдоподібності, λ, буде таким:

$$\log\lambda = log\frac{L(H_1)}{L(H_1)} = log\frac{b(c_{12},c_1,p)b(c_2-c_{12},N-c_1,p)}{b(c_{12},c_1,p_1)b(c_2-c_{12},N-c_1,p_2)}$$
$$= logL(c_{12},c_1,p) + logL(c_2-c_{12},N-c_1,p) - logL(c_{12},c_1,p_1)$$
$$- logL(c_2-c_{12},N-c_1,p_2), \qquad (16)$$

де $L(k,n,x)\ =\ xk(1\ -\ x)n\ -\ k$. А. Муд (1974) в своєму дослідженні показав, що $-2log\lambda$ має асимптотичний хі-квадрат розподіл. Отже, якщо обчислені значення $-2log\lambda$ менші за значення хі-квадрат на заданому рівні значущості, приймається нульова гіпотеза незалежності, інакше приймається гіпотеза, згідно з якою w^1w^2 є колокацією. [31]

В результаті метод логарифмічної правдоподібності дає статистику, яка говорить про те, наскільки вірогідніше одна гіпотеза, ніж інша; чим більше число, тим ближче кандидат до того щоби дійсно бути колокацією. Метод застосовується до всіх n-грамів у корпусі, і генерується впорядкований список для пошуку колокацій.

2.2.6. Метод точкової взаємної інформації

Ще одним методом для пошуку цікавих колокацій є точкова взаємна інформація . У теорії інформації взаємна інформація визначається як величина, яка вимірює взаємну залежність двох змінних. При виділенні спільного розташування замість двох випадкових величин, визначення змінюється для значень випадкових величин. Таким чином, вводиться нова міра, точкова взаємна інформація. [9]

Якщо ми запишемо x і y для першого та другого слова відповідно, тоді точкова взаємна інформація для них визначається за формулою (17).

$$I(x,y) = log_2 \frac{P(x,y)}{P(x) \cdot P(y)} \quad (17)$$

Якщо слова x та y незалежні одне від одного; ймовірність вживання слів разом повинна дорівнювати множенню їхніх власних ймовірностей $P(x,y) = P(x) \cdot P(y)$. В цьому випадку взаємна інформація буде нульовою $I(x,y) = 0$, що вказує на те, що слова не утворюють колокацію. Отже, чим далі значення взаємної інформації комбінації від нуля, тим більше вірогідність того, що ця комбінація є колокацією.

Можна сказати, що взаємна інформація є хорошим показником незалежності. Але це погана міра залежності, оскільки для залежності оцінка визначається на основі частоти окремих слів. За інших рівних умов біграми, що складаються з слів, що зустрічаються нечасто, отримають вищий бал, ніж біграми, що складаються зі слів, що можна зустріти часто. Це протилежне тому, що очікується, оскільки вища частота означає більше свідчень того, що комбінація є колокацією, саме такі комбінації повинні отримувати високий ранг. Одним із рішень, яке було запропоновано для цього, є використання фільтрації та аналізу лише слів із частотою принаймні три. Однак такий крок не вирішує основну проблему, а лише покращує результати.

Оскільки точкова взаємна інформація не дуже добре відображає інтуїтивне уявлення про цікаву колокацію, вона не дуже часто використовується для рішення практичних завдань або перевизначається як $C(w^1w^2)I(w^1;w^2)$ для компенсації зміщення вихідного визначення на користь подій з низькою частотою.

2.3. Експериментальна оцінка методів пошуку словосполучень

Пошук словосполучень зазвичай виконується на основі корпусу, тобто в ідеалі потрібен корпус з розмітками словосполучень, але для української мови немає настільки ж зручних текстових корпусів, як для англійської. Через великі розміри корпусів неможливо вручну позначити всі словосполучення в корпусі. У багатьох дослідженнях автори витягують базовий набір із корпусів і реалізують методи на цьому наборі. Базовий набір може бути побудований різними способами, наприклад, його можна побудувати з певного словосполучення з урахуванням частини мови, так пари прикметник-іменник у корпусах можуть бути обрані для створення базового набору, як у дослідженні Еверта та Кренн [7], або набір можна отримати зі словника [17].

В цьому дослідженні був використаний інший підхід для створення базового набору. Оскільки ціллю дослідження є побудова словника словосполучень української мови, на основі необроблених текстових даних, в даному експерименті не використовувалися корпуси текстів з розміткою словосполучень, а були проаналізовані необроблені текстові дані. Спочатку були виконані кроки попереднього оброблення текстів, після цього п’ять методів, визначених у попередньому розділі, застосовуються для створення ранжованого списку біграм. У рейтингових списках кандидати, які набрали найвищі бали, вважаються колокаціями. Таким чином, стало можливим порівняння всіх методів на основі одного і того ж набору даних.

У цьому дослідженні використовувався корпус текстів українською мовою під назвою UA-GEC, який є першим анотованим корпусом саме українською мовою. Корпус розроблений компанією Grammarly та містить в собі тексти різних жанрів, авторів, сфер наукової та суспільної діяльності. Всього корпус налічує 328771 слів, 20715 речень та містить тексти 492 авторів. Для зручного користування корпусом компанією був також розроблений Python пакет. Тексти в корпусі містяться в двох розділах – навчальному та тестовому. Для проведення експерименту були використані обидва розділи і в сумі було проаналізовано 166330 слів. [26]

Оскільки словосполучення визначаються як комбінації слів, що часто зустрічаються, було виключено сполучення слів, які зустрічалися менше трьох разів у корпусі.

2.3.1. Метод оцінки результатів

Оцінка методів пошуку, описаних у попередніх пунктах, виконується за допомогою обчислення значень влучності та повноти, які часто використовуються як показники ефективності пошуку інформації. Для пошуку колокацій повноту можна визначити як частку колокацій у корпусі або базовому наборі, який успішно знайдені. Влучність — це частка справжніх колокацій у отриманому списку кандидатів. Позначивши ϵ як колокації, виділені з базового набору, а δ як кількість істинних колокацій у базовому наборі, влучність та повноту можна визначити як

$$r = \frac{|\epsilon \cap \delta|}{|\delta|} \tag{18}$$

$$p = \frac{|\epsilon \cap \delta|}{|\epsilon|} \tag{19}$$

При представленні значень влучності та повноти був застосований підхід Еверта та Кренн, який передбачає в формуванні випадкової вибірки невеликого розміру, в яку будуть входити позитивні та негативні приклади, тобто комбінації слів, що є словосполученнями та ні відповідно. [7]

2.3.2. Експериментальна оцінка ефективності мір асоціації

Перед початком експерименту були завантажені тексти з корпусу та проведена нормалізація тексту. Були виконані кроки, які описані в пункті 2.1. Для проведення експерименту була використана Python бібліотека NLTK, яка є найпопулярнішою бібліотекою для досліджень та додатків, де використовуються

NLP алгоритми. В даній бібліотеці є імплементація описаних методів пошуку словосполучень. [32]

Отже, в ході експерименту тексти корпусу UA-GEC біли проаналізовані на предмет наявності словосполучень, які складаються з двох та трьох слів. На Рисунку 12 представлені 20 біграм, які мають найвищий рейтинг. Результати розділені за методом, за допомогою якого вони були отримані. На Рисунку 13 Можна побачити аналогічні результати, тільки для словосполучень з трьох слів.

	Frequency	PMI	T-test	Chi-Sq Test	Likeihood Ratio Test
0	(мою, думку)	(потерпимо, зневаги)	(мою, думку)	(поцілувався, страшненькою)	(мою, думку)
1	(штучного, інтелекту)	(потужно, інформувати)	(штучного, інтелекту)	(посідала, територіальної)	(штучного, інтелекту)
2	(точки, зору)	(потужним, ультрафіолетом)	(точки, зору)	(посіви, знищуються)	(точки, зору)
3	(наступного, дня)	(потугу, руїнної)	(наступного, дня)	(посунулась, милю)	(ніна, іванівна)
4	(має, право)	(потрошки, поверталось)	(має, право)	(посуваються, встаючи)	(молекулярної, динаміки)
5	(ніна, іванівна)	(потребую, старту)	(ніна, іванівна)	(постійною, міграцією)	(сталого, розвитку)
6	(сталого, розвитку)	(потребувало, гнучкої)	(сталого, розвитку)	(постійних, відрядженнях)	(державною, мовою)
7	(державною, мовою)	(потрапляю, висловлюю)	(державною, мовою)	(постулати, недооцінені)	(надання, впевненості)
8	(останнім, часом)	(потраплю, ідеальну)	(останнім, часом)	(постулат, виправданий)	(наступного, дня)
9	(дає, змогу)	(потонули, земному)	(дає, змогу)	(пострілу, снайперської)	(своєю, чергою)
10	(теорії, ігор)	(потомства, розрада)	(надання, впевненості)	(постригся, ченці)	(маркіяна, шашкевича)
11	(своєю, чергою)	(потицяти, віртуалці)	(своєю, чергою)	(поставлять, крапельницю)	(суспільно, орієнтоване)
12	(іншого, боку)	(потилиця, стиснена)	(теорії, ігор)	(посприяло, активній)	(суспільно, орієнтованого)
13	(надання, впевненості)	(потенціали, згладжують)	(іншого, боку)	(посприяли, зникненню)	(останнім, часом)
14	(сказав, скрудж)	(пострілу, снайперської)	(сказав, скрудж)	(посполитих, посполиті)	(другої, світової)
15	(теорія, ігор)	(потемніли, примарилось)	(молекулярної, динаміки)	(посольстві, культурний)	(дає, змогу)
16	(пів, години)	(потаповича, скупого)	(другої, світової)	(послідовний, скований)	(теорії, ігор)
17	(другої, світової)	(потайний, замкнутий)	(теорія, ігор)	(послуговуватися, транспортним)	(усунення, неоднозначності)
18	(молекулярної, динаміки)	(посіяло, насіння)	(пів, години)	(послана, филипа)	(ямної, культури)
19	(двадцять, років)	(посідала, територіальної)	(маркіяна, шашкевича)	(поскакушками, радували)	(ніни, іванівни)

Рисунок 12 – Порівняння перших 20 біграм знайдених розглянутими методами

	Frequency	PMI	T-test	Chi-Sq Test	Likeihood Ratio Test
0	(суспільно, орієнтоване, навчання)	(*навчання, провадити, кейсовим)	(суспільно, орієнтоване, навчання)	(*навчання, провадити, кейсовим)	(суспільно, орієнтоване, навчання)
1	(звітності, сталого, розвитку)	(поповнилося, залізницею, тунелем)	(суспільно, орієнтованого, навчання)	(поповнилося, залізницею, тунелем)	(суспільно, орієнтованого, навчання)
2	(суспільно, орієнтованого, навчання)	(порішав, затащили, катки)	(звітності, сталого, розвитку)	(порішав, затащили, катки)	(мою, думку, людина)
3	(двадцять, років, довгий)	(порізав, електричну, проводку)	(двадцять, років, довгий)	(порізав, електричну, проводку)	(мою, думку, бурхлива)
4	(аудиту, звітності, сталого)	(поріжте, кубиками, ребром)	(всередині, вуглецевих, нанотрубок)	(поріжте, кубиками, ребром)	(мою, думку, прояви)
5	(сказав, високий, чоловік)	(порядне, ґаздівство, остапову)	(аудиту, звітності, сталого)	(порядне, ґаздівство, остапову)	(мою, думку, озвучкою)
6	(другої, світової, війни)	(поршні, заскрипіли, штовхнули)	(другої, світової, війни)	(поршні, заскрипіли, штовхнули)	(мою, думку, найефективнішим)
7	(всередині, вуглецевих, нанотрубок)	(поршневих, шиберних, відцентрової)	(води, всередині, вуглецевих)	(поршневих, шиберних, відцентрової)	(мою, думку, порушується)
8	(води, всередині, вуглецевих)	(порцію, демотивації, вивчи)	(сказав, високий, чоловік)	(порцію, демотивації, вивчи)	(лінивих, мою, думку)
9	(ресурсів, черпаєш, корисну)	(портів, танкє, горєл)	(ресурсів, черпаєш, корисну)	(портів, танкє, горєл)	(епатажу, мою, думку)
10	(черпаєш, корисну, інформацію)	(портупеї, меча, зашнурованому)	(користувацького, інтерфейсу, державною)	(портупеї, меча, зашнурованому)	(дидероти, мою, думку)
11	(одному, міських, департаментів)	(портативними, лікарнями, медиками)	(черпаєш, корисну, інформацію)	(портативними, лікарнями, медиками)	(акторами, мою, думку)
12	(користувацького, інтерфейсу, державною)	(порода, німецька, вівчарка)	(таблиці, каскадних, стилів)	(порода, німецька, вівчарка)	(мою, думку, ключовий)
13	(яких, ресурсів, черпаєш)	(поразкам, захоплюватися, подвигами)	(бобе, чикаго, думає)	(поразкам, захоплюватися, подвигами)	(романтика, мою, думку)
14	(старі, добрі, часи)	(порадитися, тамтешніми, інженерами)	(мер, сидячи, ліжку)	(порадитися, тамтешніми, інженерами)	(символічні, мою, думку)
15	(інтерфейсу, державною, мовою)	(попереджає, пластиковій, тарі)	(одному, міських, департаментів)	(попереджає, пластиковій, тарі)	(мою, думку, наступна)
16	(бобе, чикаго, думає)	(посиджу, втиснулась, покривало)	(інтерфейсу, державною, мовою)	(посиджу, втиснулась, покривало)	(мою, думку, правильного)
17	(учасників, бойових, дій)	(поодиноких, беріз, чагарників)	(старі, добрі, часи)	(поодиноких, беріз, чагарників)	(творчість, мою, думку)
18	(часи, двоє, чоловіків)	(пообідати, відкритій, терасі)	(часи, двоє, чоловіків)	(пообідати, відкритій, терасі)	(досвід, мою, думку)
19	(таблиці, каскадних, стилів)	(помішувала, курячий, бульйон)	(яких, ресурсів, черпаєш)	(помішувала, курячий, бульйон)	(філософське, мою, думку)

Рисунок 13 – Порівняння перших 20 біграм знайдених розглянутими методами

Поверхневий огляд результатів показує, що критерій хі-квадрат та метод взаємної інформації показали досить хороші результати щодо пошуку цікавих колокацій. Результати отримані цими методами також досить схожі, але для виділення біграм в даному корпусі кращі результати демонструє метод взаємної інформації. Результати отримані за допомогою t-критерію та методу простого підрахунку частот також схожі.

Хоча на основі отриманих результатів можна зробити певний попередній висновок, все ще не зрозуміло який метод є ефективніший для даного набору текстових даних. Адже проаналізувати всі виділені кандидати у колокації вручну дуже складно, для порівняння ефективності методів був застосований підхід Еверта та Кренн. [7]

Отже, в даному дослідження в якості позитивної та негативної вибірки були обрані по 229 кандидатів у колокації. Вибірка позитивних прикладів була сформована шляхом ручної фільтрації людиною випадково вибраних n-грамів з корпусу. При аналізі реальних даних звісно вибірка буде відрізнятися, тому що

скоріш буде більше негативних прикладів, ніж позитивних, але така тестова вибірка є достатньою для того, щоб отримати уявлення про ефективність методів та порівняти їх між собою.

Графік влучності та повноти для словосполучень, які складаються з двох слів, можна побачити на Рисунку 14. На графіку видно, що всі міри асоціації, які порівнюються, працюють краще, ніж простий підрахунок частоти, тобто використання мір асоціації є виправданим та ефективним. Можна також побачити, що метод поточкової взаємної інформації працює найкраще, другий хі-квадрат, за ним логарифмічна правдоподібність. Гірші результати показав t-критерій.

На Рисунку 15 також можна побачити графік влучності та повноти але вже для словосполучень з трьох слів. На цьому графіку видно, шо й для триграмів метод поточкової взаємної інформації показав найкращі результати, але цього разу дуже близький до нього виявився критерій хі-квадрат. Логарифмічна правдоподібність цього разу показала приблизно такі ж результати, як простий підрахунок частоти та t-критерій. Отже, можна стверджувати, що для дослідження колокацій різної довжини, різні методи можуть працювати по-різному.

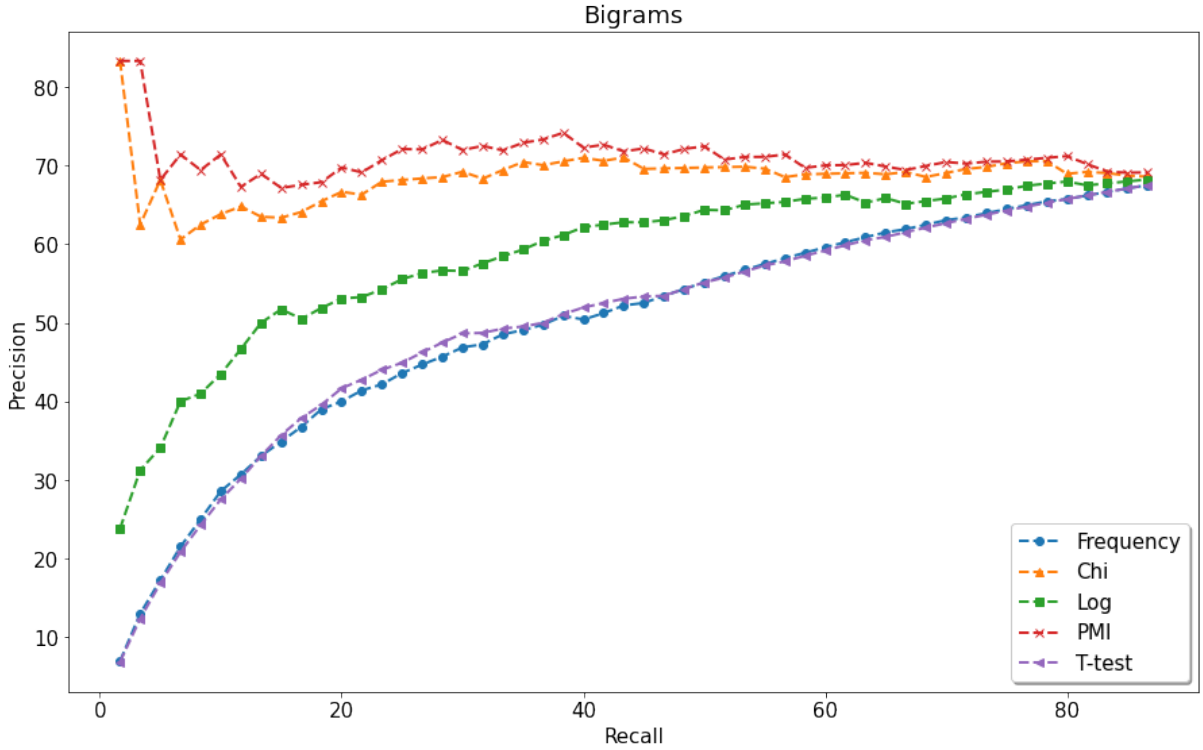


Рисунок 14 – Графік влучності та повноти для біграм

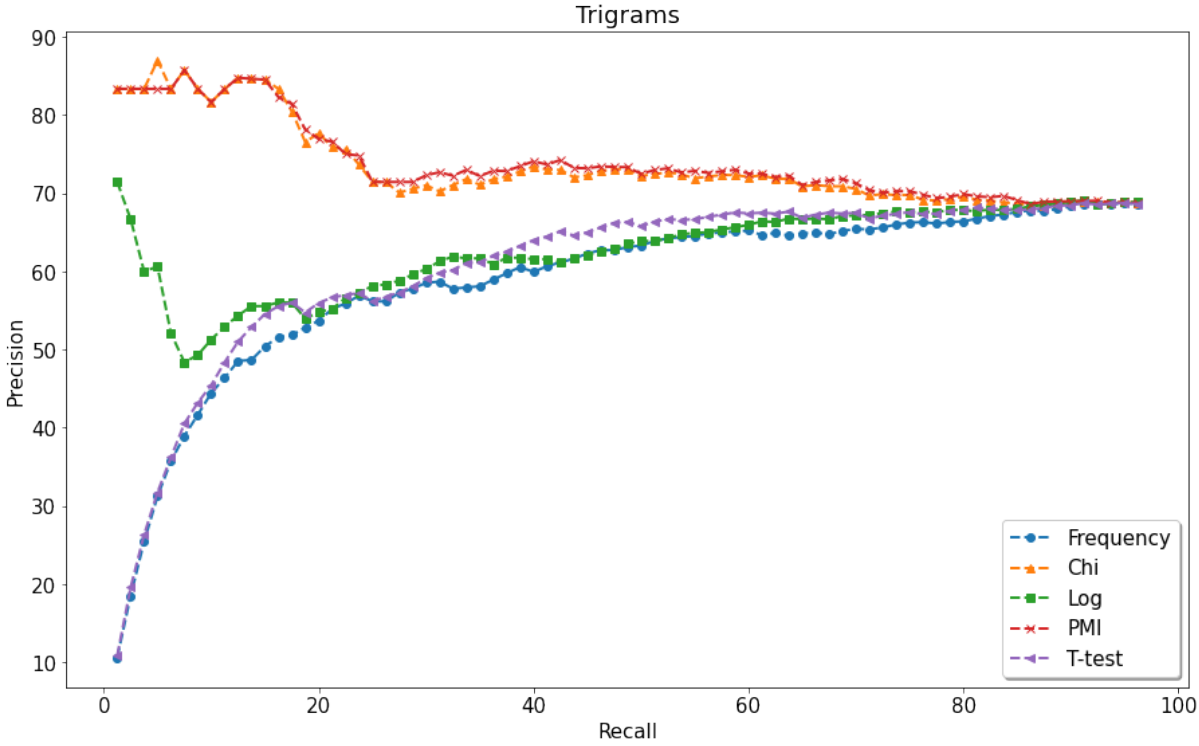


Рисунок 15 – Графік влучності та повноти для триграм

З отриманих результатів можна зробити висновок, що для пошуку словосполучень в необроблених текстових даних з метою побудови словника словосполучень української мови, можна обрати метод поточкової взаємної інформації, оскільки на графіках видно, що він показав найкращі результати. Окрім цього можна поєднати результати роботи різних методів, які показали свою ефективність для досліджуваного корпусу текстів. Можна сказати, що поєднання колокацій знайдених методом поточкової взаємної інформації та методом логарифмічної правдоподібності, дасть хороші результати.

2.3.3. Фільтрація результатів за допомогою розмічування частин мови

Підрахунок частоти є дуже простим методом, але цікаво подивитися, як покращує результат якийсь вид фільтрації, наприклад, на основі частини мови. Джастесон і Кац запропонували деякі шаблони розмічування частин мови, які

можна використовувати для ефективної фільтрації кандидатів у колокації. Дані шаблони були представлені в Таблиці 2.1.

На Рисунку 16 представлені 10 найкращих кандидатів у колокації до фільтрації з використанням шаблонів розмічування частин мови, а на Рисунку 17 вже після. Можна побачити, що були відфільтровані деякі некоректні варіанти, наприклад «сказав високий чоловік» не є словосполученням, оскільки в ньому присутні підмет та присудок, а значить це повноцінне речення. Також кандидати сталі більш цікавими з точки зору лінгвістики, оскільки з більшості кандидатів після фільтрації можна відновити контекст та словосполучення несуть певний зміст.

	trigram	freq
39972	(суспільно, орієнтоване, навчання)	12
10900	(звітності, сталого, розвитку)	11
37783	(суспільно, орієнтованого, навчання)	11
16570	(двадцять, років, довгий)	7
10899	(аудиту, звітності, сталого)	6
18540	(під, другої, світової)	6
15734	(yevhenii, kanivets, blog)	6
20819	(сказав, високий, чоловік)	6
43073	(співбесіда, найбільше, запам)	6
18677	(другої, світової, війни)	6

Рисунок 16 – Найкращі 20 кандидатів до фільтрації за шаблонами розмічування частин мови

	trigram	freq
10900	(звітності, сталого, розвитку)	11
10899	(аудиту, звітності, сталого)	6
18677	(другої, світової, війни)	6
8765	(води, всередині, вуглецевих)	6
43112	(ресурсів, черпаєш, корисну)	5
21984	(користувацького, інтерфейсу, державною)	5
19120	(старі, добрі, часи)	5
21985	(інтерфейсу, державною, мовою)	5
6478	(часи, двоє, чоловіків)	5
139516	(таблиці, каскадних, стилів)	5

Рисунок 17 – Найкращі 20 кандидатів після фільтрації за шаблонами розмічування частин мови

Для більш наочного порівняння було побудовано графік влучності та повноти для методу хі-квадрат з фільтрацією результатів за допомогою розмічування частин мови та без. Його можна побачити на Рисунку 18.

На графіку видно, шо кандидати , які були відфільтровані, мають більшу влучність, отже, можна стверджувати, що фільтрація кандидатів за допомогою розмічування частин мови дійсно вдосконалює статистичні методи пошуку словосполучень.

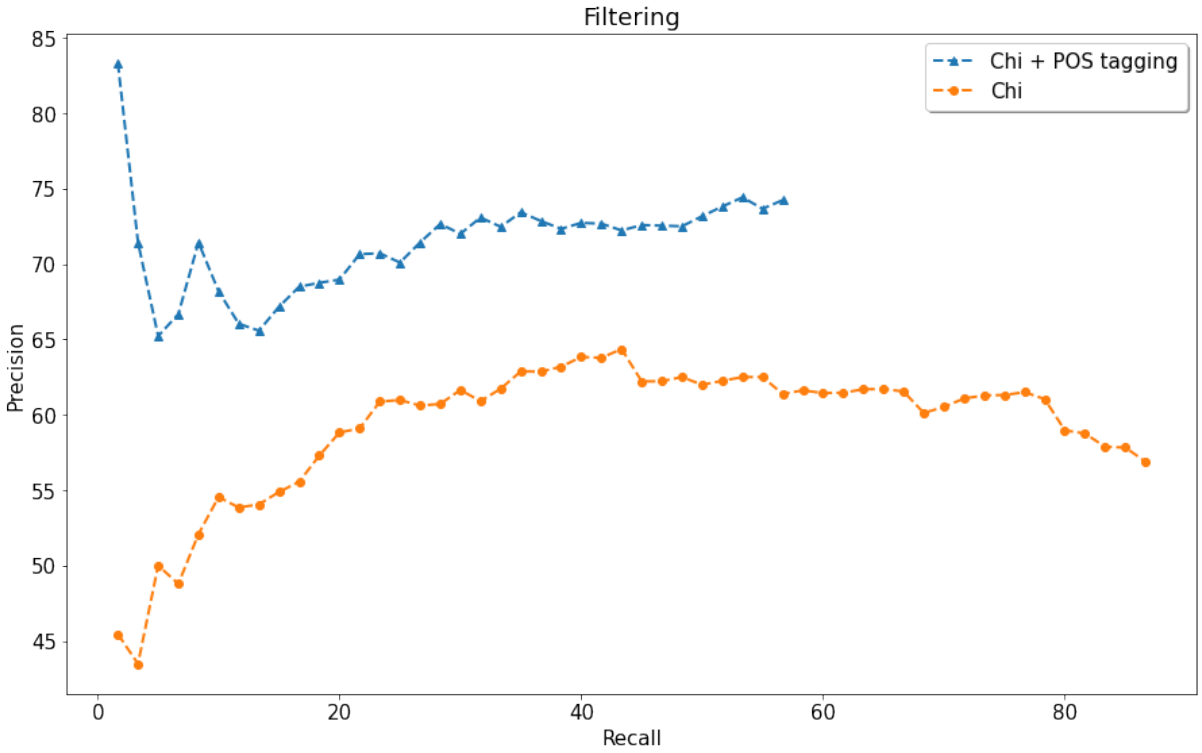


Рисунок. 18 – Графік влучності та повноти для методу хі-квадрат з розмічуванням за частинами мови та без

Висновки до другого розділу

В даному розділі були визначені необхідні кроки для попереднього оброблення текстових даних, що повинно покращити результати роботи NLP алгоритмів. Були визначені наступні кроки: видалення повторюваних пробілів, пунктуації, посилань та номерів; виправлення орфографії; стемінг та лематизація; видалення стоп-слів. Був також проведений експеримент, який показав ефективність проведення нормалізації тексту, для роботи з алгоритмами NLP.

Також у даному розділі були розглянуті статистичні методі знаходження та ранжування кандидатів у словосполучення. Розглянуті наступні методи: простий підрахунок частот, t-критерій Стьюдента, критерій хі-квадрат, логарифмічна правдоподібність та метод взаємної інформації. Був також проведений експеримент і визначено, який метод працює найефективніше для текстів саме українською мовою. Найефективнішим методом виявився метод взаємної інформації. Також визначено, що різні методи дають різні результати, які є також досить точними, а отже на практиці методи можна комбінувати.

Експериментальним шляхом визначено, що знайдені кандидати у словосполучення можна також відфільтрувати за частинами мови, що покращить результати – ті словосполучення, які залишаться після фільтрації стануть більш схожими на природну мову.

3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ СЛОВНИКА СЛОВОСПОЛУЧЕНЬ

3.1. Опис використаних інструментів, мови та сторонніх залежностей

На основі досліджень проведених та описаних в попередніх розділах роботи була розроблена база даних словосполучень української мови.

Для розроблення описаного програмного забезпечення була обрана мова програмування Python. У Python є багато речей, які роблять його дійсно хорошим вибором мови програмування для проекту, який реалізовує NLP алгоритми. Простий синтаксис і прозора семантика цієї мови роблять її чудовим вибором для проектів, які включають завдання обробки природної мови. Крім того, розробники можуть користуватися чудовою підтримкою для інтеграції з іншими мовами та інструментами, які стануть у пригоді для таких методів, як машинне навчання. Окрім цього в цій універсальній мові програмування є ще одна особливість, яка робить її дуже зручною технологією для оброблення природної мови. Python надає розробникам велику колекцію інструментів і бібліотек NLP, які дозволяють розробникам легко вирішувати велику кількість завдань, пов’язаних з даною спеціалізацією, таких як класифікація документів, моделювання тем, розмічування частин мови (POS), вектори слів та аналіз настроїв.

У минулому лише експерти могли брати участь у проектах оброблення природної мови, які вимагали вищих знань з математики, машинного навчання та лінгвістики. Тепер розробники можуть використовувати готові інструменти, які спрощують попередню обробку тексту, щоб вони могли зосередитися на створенні моделей машинного навчання.

Серед відомих бібліотек NLP для Python є такі як: Natural Language Toolkit (NLTK), CoreNLP, Gensim, spaCy та багато інших. В даній роботі була вибрана бібліотека NLTK, так як це популярна бібліотека, яка підтримує такі завдання, як класифікація, визначення коренів, розмічування за частинами мови, синтаксичний аналіз та токенізацію в Python. По суті, це основний інструмент для обробки природної мови та машинного навчання.

Бібліотека була розроблена Стівеном Бердом і Едвардом Лопером з Університету Пенсильванії і зіграла ключову роль у проривних дослідженнях в сфері NLP.

NLTK є провідною платформою для створення програм Python для роботи з даними природної мови, яка надає прості у використанні інтерфейси для більш ніж 50 корпусів і лексичних ресурсів, таких як WordNet, а також набір бібліотек оброблення тексту для основних задач NLP, та активну спільноту розробників. Завдяки вичерпній документації, NLTK підходить для лінгвістів, інженерів, студентів, викладачів, дослідників та інших, що працюють в цій галузі. [32]

Бібліотека NLTK була використана для попереднього оброблення текстів, а саме токенізації тексту. Також дана бібліотека містить методи, які реалізують міри асоціації, що використовуються для виділення словосполучень з текстових даних.

Ще одним використаним інструментом став морфологічний аналізатор pymorphy2. Ця бібліотека була використана для отримання леми слова при нормалізації тексту, та для того, щоб повернути граматичну інформацію про слова, які складають словосполучення. Словники для pymorphy2 поставляються окремо, тому був також встановлений словник pmorphy2-dicts-uk – словник української мови. [33]

Іншим кроком нормалізації у даному дослідженні було визначено виправлення орфографії. Для вирішення цієї задачі був використаний hunspell, як описано в попередньому розділі 2.

Оброблення великої кількості текстових даних є ресурсоємним процесом та потребує значної кількості часу. Для прискорення роботи цих алгоритмів був використаний фреймворк для розподілених обчислень Apache Spark. Для роботи зі Spark був обраний програмний пакет PySpark. Він не тільки дозволяє писати Spark додатки за допомогою API Python, але й надає інтерактивну оболонку PySpark для зручного аналізу даних у розподіленому середовищі. PySpark підтримує більшість функцій Spark, таких як Spark SQL, DataFrame, Streaming, MLlib, тобто функції машинного навчання, і Spark Core.

В даному дослідженні були використані наступні функції та переваги PySpark:

- Spark SQL — це модуль Spark для оброблення структурованих даних, що забезпечує абстракція програмування під назвою DataFrame, а також може діяти як розподілений механізм запитів SQL.
- Spark Core — модуль, що є базовим механізмом загального виконання для платформи Spark, на якому побудовані всі інші функції, надає стійкий розподілений набір даних (англ. Resilient Distributed Dataset, RDD) і обчислювальні можливості в пам'яті. [34]

Ще одною перевагою PySpark є те, що його API схоже з API дуже популярної бібліотеки Pandas, тому мати єдину кодову базу, яка працює як із Pandas. так і зі Spark дуже зручно, тому що можна уніфікувати тести, набори даних, тощо. [35]

Для того, щоб зберігати та виконувати пошук за отриманими результатами аналізу текстових даних на предмет знаходження словосполучень, було використане сховище даних Elasticsearch, яке дозволяє швидко зберігати великі обсяги даних та будувати складні запити для пошуку у збережених даних, збирати статистику за цими даними, виконувати складні агрегації даних, та що не менш важливо – ця система добре масштабується. [36]

Для адміністрування індексів Elasticsearch та можливості виконувати пошукові запити для більш детального аналізу отриманих результатів була використана система Kibana — безкоштовний відкритий інтерфейс користувача, який дозволяє візуалізувати дані з Elasticsearch та виконувати менеджмент Elastic Stack. [37]

Розроблювана система представляє собою веб-додаток, тому для реалізації його API було використано мікрофреймворк Flask. «Мікро» означає, що Flask прагне зберегти ядро додатку простим, але розширюваним, тому Flask легко налаштовувати так, як потрібно під вирішення певних задач, таким чином можна створити легку систему, яка не має ніяких зайвих елементів, тільки ті, що потрібні для вирішення певної задачі.

За замовчуванням Flask не включає рівень абстракції бази даних, перевірку форм та інше, де вже існують різні бібліотеки, які можуть це обробляти. Натомість Flask підтримує розширення для додавання такої функціональності до програми так, якби вона була реалізована в самому Flask. Численні розширення забезпечують інтеграцію бази даних, перевірку форм, обробку завантаження, різні технології відкритої аутентифікації тощо. [38]

Для відображення шаблону веб-сторінки завантаження вхідних текстових даних був використаний механізм шаблонів Jinja2.

Jinja2 є одним з найбільш використовуваних механізмів шаблонів для Python. Розробники системи надихнулись системою шаблонів Django, але розширили її виразною мовою, що надає розробникам шаблонів потужніший набір інструментів. [39]

3.2. Опис архітектури розробленого програмного забезпечення

Розроблене програмне забезпечення генерації бази даних словосполучень української мови представляє собою веб-додаток, а саме бекенд систему. Даний веб-додаток має тип архітектури Model-View-Controller (MVC).

MVC — це широко використовуваний архітектурний шаблон програмного забезпечення, традиційно він використовувався для розроблення десктопних програм з графічним інтерфейсом користувача, але згодом став популярним для розроблення веб-додатків. Багато мов програмування мають фреймворки MVC, які полегшують реалізацію шаблону. Цей шаблон має три компоненти, а саме модель, представлення та контролер.

- Модель – центральний компонент шаблону, який представляє дані програми, також представляє основну бізнес-логіку, яка діє на ці дані, при цьому не знає стану представлення чи контролера, тому коли дані в моделі змінюються, вона просто повідомляє своїх слухачів про цю зміну.

- Представлення – є інтерфейсом користувача та відповідає за відображення поточного стану моделі, а також надає засіб для взаємодії користувача з програмою.
- Контролер – приймає вхідні дані та перетворює їх в команди для змін в моделі або представленні, тобто дозволяє користувачу взаємодіяти з програмою, реагує на введення користувача та отримує вхідні дані, за бажанням перевіряє їх, а потім передає вхідні дані моделі.

Незважаючи на те, що MVC спочатку був розроблений для десктопних додатків, він широко використовується як архітектурний шаблон веб-додатків в популярних мовах програмування. Також існує багато веб-фреймворків, які забезпечують реалізацію шаблону. Ці програмні фреймворки відрізняються за своїми інтерпретаціями, головним чином у тому, як відповідальність MVC розподіляється між клієнтом і сервером.

Деякі веб-фреймворки MVC використовують підхід тонкого клієнта, який розміщує майже всю логіку моделі, представлення та контролера на сервері. У цьому підході клієнт надсилає або запити, або форми до контролера, а потім отримує повну та оновлену веб-сторінку або інший документ із представлення, а модель повністю існує на сервері.

В даній роботі при розробленні додатку саме використовується підхід тонкого клієнта, оскільки вся робота додатку виконується на серверній частині.

Бізнес логіка додатку поділяється на два великих модулі – це модуль оброблення вхідних текстових даних, тобто наповнення бази даних інформацією, яка отримується за допомогою оброблення вхідних текстових даних з використанням розробленого NLP пайплайну, та модуль, який відповідає за оброблення запитів користувача на пошук словосполучень, в які входить слово, що подане у вхідних параметрах запиту.

Загальну схему розробленої системи можна побачити на Рисунку 19.

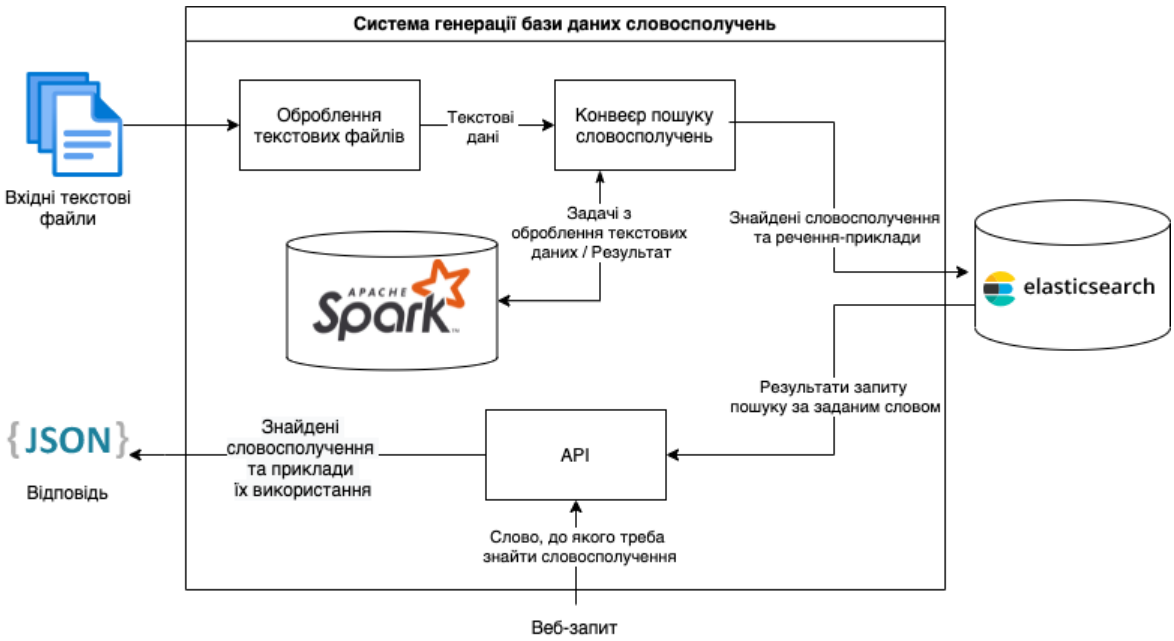


Рисунок 19 – Схема розробленої системи генерації бази даних словосполучень

3.3. Особливості реалізації модулю виділення словосполучень із текстових даних

Даний модуль за логікою поділяється на три частини. Перша – це попередня обробка вхідних текстових даних, друга – це саме виділення словосполучень із підготовлених даних за допомогою обраного статистичного методу – взаємної інформації, а третя – фільтрація знайдених кандидатів у словосполучення. Схема даного модулю зображена на Рисунку 20.

Рисунок 20 – Схема модулю виділення словосполучень з текстових даних

3.3.1. Особливості реалізації нормалізації вхідних текстових даних

Для попередньої обробки текстових даних реалізовані кроки, які були описані в пункті 2.1, а саме видалення повторюваних пробілів, пунктуації,

посилань та номерів, виправлення орфографії, стемінг та лематизація і видалення стоп-слів.

Кроки нормалізації, такі як, видалення URL-посилань, знаків пунктуації, нормалізація пробілів та видалення номерів було зроблено за допомогою регулярних виразів, а токенізація слів у реченні проведена за допомогою методу бібліотеки NLTK – *nltk.word_tokenize*.

Далі був реалізований крок виправлення орфографії слів. Як було описано в пункті 2.1 для цього була використана бібліотека Hunspell. На Рисунку 21 можна побачити блок-схему алгоритму перевірки та виправлення орфографії слів, також нижче наведений Лістинг 1 методу, що реалізує цей алгоритм. Спочатку слово перевіряється на правильність написання за допомогою методу бібліотеки *hunspell.spell*. Цей метод приймає в якості аргумента слово, яке треба перевірити, та повертає значення булевого типу.

Лістинг 1. Метод виправлення орфографії слова

normalize.py

```
def correct_words(hunspell, words):
    corrected = []
    for w in words:
        ok = hunspell.spell(w)
        if not ok:
            suggestions = hunspell.suggest(w)
            if len(suggestions) > 0:
                best = suggestions[0]
                corrected.append(best)
            else:
                corrected.append(w)
        else:
            corrected.append(w)
    return corrected
```

Якщо повернуте значення дорівнює False, і слово написане неправильно, то за допомогою методу *suggest*, отримуються варіанти правильного написання даного слова, якщо такі варіанти знайдені, тобто їх кількість більше нуля, то обираємо найкращий з них і повертаємо замість вхідного слова. Якщо ж слово спочатку було написано правильно кроки з підбором варіантів написання пропускаються і повертається початковий варіант слова.

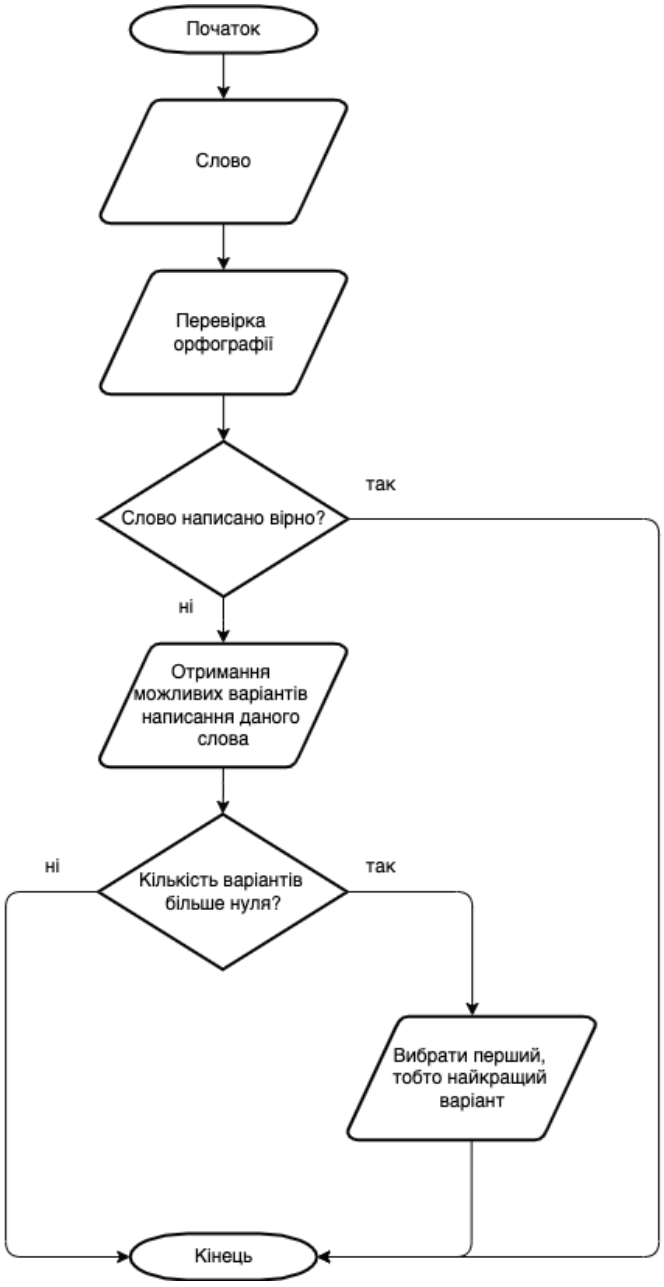


Рисунок. 21 – Алгоритм перевірки та виправлення орфографії слова

Наступним кроком нормалізації стало видалення стоп-слів. Цей крок був реалізований простою перевіркою слів на входження до завчасно заготованого списку шумових слів. В такий список входять слова, що складаються з однієї букви, займенники, прийменники, дієприкметники та ін. Ще одним кроком нормалізації стала лематизація слів, яка також була реалізована за допомогою бібліотеки Hunspell, а саме за допомогою методу *hunspel.stem*.

3.3.2. Особливості реалізації виділення словосполучень із нормалізованих текстових даних

Після цього нормалізовані текстові дані готові бути обробленими статистичним методом для знаходження кандидатів у словосполучення. Ця частина системи реалізована за допомогою класів модулю бібліотеки NLTK – *nltk.collocations.* Пакет *collocation* надає засоби пошуку кандидатів у словосполучення, які за замовчуванням розглядають усі n-грами в тексті у якості кандидатів. Наприклад, для знаходження словосполучень, що складаються з двох слів, використовується клас *BigramCollocationFinder.* Для завантаження даних, що будуть проаналізовані, використовується метод класу *from_words.* За імплементацію міри асоціації, яка присвоїть кожному знайденому кандидату певну оцінку відповідає клас *nltk.collocations.BigramAssocMeasures.* На Лістингу 2, що наведений нижче можна побачити фрагмент коду, який реалізує знаходження кандидатів у словосполучення, аналізуючи список лем, що обули отримані в результаті нормалізації вхідних текстових даних.

У прикладі можна побачити, що був ініціалізований клас *BigramAssocMeasures,* а потім і *BigramCollocationFinder,* у який були завантажені дані для аналізу – список нормалізованих слів *flat_lemmas_list*. Для ранжування даних був застосований метод взаємної інформації, і результати збережені у Pandas DataFrame, відсортовані по порядку спадання оцінки, що була присвоєна їм статистичним методом.

Лістинг 2. Фрагмент коду, що відповідає за виділення словосполучень

```
ngrams.py

ngrams = nltk.collocations.BigramAssocMeasures()
ngrams_finder = nltk.collocations.BigramCollocationFinder.from_words(
    flat_lemmas_list,
)
ngrams_finder.apply_word_filter(lambda w: len(w) < 3)
ngrams_pmi_table = pd.DataFrame(
    list(ngrams_finder.score_ngrams(ngrams.pmi)),
    columns=['bigram', 'PMI'],
).sort_values(by='PMI', ascending=False)
```

3.3.3. Особливості реалізації фільтрації отриманих кандидатів

Отримані після використання статистичних методів до нормалізованих даних кандидати впорядковуються за значенням оцінки, яку їм присвоїв метод взаємної інформації. Як було показано в пункті 2.3, фільтрація словосполучень може значно покращити отримані результати та відкинути ті кандидати, які не є словосполученнями. Був обраний спосіб фільтрації за розмічуванням частин мови, тобто слова у словосполученні повинні відповідати визначеним шаблонам словосполучень.

Частина мови кожного слова у словосполучення визначена за допомогою бібліотеки pymorphy2. За допомогою методу *parse* можна отримати граматичну інформацію про слово, а саме тег, який позначає до якої частини мови відноситься слово. Визначені теги порівнюються із визначеними наборами допустимих частин мов. Якщо кандидат відповідає одному з визначених шаблонів, то він приймається як словосполучення і залишається у списку результатів, якщо ж ні – видаляється, оскільки не є словосполученням.

3.3.4. Структура та індексація даних у сховище даних

Як раніше було визначено, в якості сховища даних був обраний Elasticsearch – сховище даних, яке дуже добре підходить для завдань, де потрібно швидко зберігати великі обсяги даних та виконувати повнотекстовий пошук.

Всі дані, які були отримані в результаті аналізу вхідних текстових даних зберігаються у DataFrame звідки завантажуються в Elastic індекс.

Індекс в Elasticsearch схожий на «базу даних» у реляційній базі даних. Він має відображення (англ. mapping), яке визначає типи, які відповідають даним, що зберігаються в індексі. Тобто індекс — це логічний простір імен, який зіставляється з одним або кількома основними сегментами і може мати нуль або більше фрагментів реплік. Отже, індекс визначає два поняття. По-перше, індекс — це

певний тип механізму організації даних, що дозволяє користувачеві розділяти дані певним чином, по-друге, концепція стосується реплік і фрагментів, механізму, який Elasticsearch використовує для розподілу даних по кластеру. В розробленому додатку був створений індекс під назвою *ngrams,* представлення якого наведено на Лістингу 3.

Лістинг 3. Представлення індексу Elasticsearch – ngrams

mapping.json

```
{
  "mappings": {
    "_doc": {
      "properties": {
        "id": {
          "type": "text",
          "fields": {
            "keyword": {"type": "keyword"}
          }
        },
        "ngram": {
          "type": "text",
          "fields": {
            "keyword": {"type": "keyword"}
          }
        },
        "pos": {
          "type": "text",
          "fields": {
            "keyword": {"type": "keyword"}
          }
        },
        "sentence": {
          "type": "text",
          "fields": {
            "keyword": {"type": "keyword"}
          }
        }
      }
    }
  }
}
```

В даному індексі визначений тип *_doc*, що представляє кожний запис в індексі та містить наступні поля:

- *id* – унікальний ідентифікатор документу,
- *ngram* – поле, яке містить саме словосполучення,
- *pos* – поле, що містить розмітки частин мови слів, що входять у словосполучення,

- *sentence* – поле, в якому знаходиться речення, в якому було знайдене словосполучення, що знаходиться у полі *ngram,* його потрібно зберігати в документі поряд зі словосполученням, оскільки воно може слугувати прикладом вживання цього словосполучення у природному мовленні.

Можна також побачити, що кожне поле документу має свій тип. В даному випадку всі поля мають тип *text,* але Elasticsearch надає досить велику кількість типів для різних даних, під різні потреби.

Для індексування даних DataFrame, в якому вони зберігаються перетворюється на JSON дані та bulk-запитом відправляється на завантаження в Elasticsearch індекс.

Отже, після нормалізації, знаходження словосполучень у текстових даних, ранжування за допомогою методу взаємної інформації та фільтрації за частинами мови, дані потрапляють до Elasticsearch у визначеному вище форматі, та зберігаються там для подальшого використання цих даних в API пошуку.

3.4. Прискорення оброблення текстових даних

Оскільки алгоритми нормалізації та виділення словосполучень працюють недостатньо швидко, одним з основних завдань даного дослідження є прискорити роботу даних алгоритмів. Для вирішення цього завдання був розроблений альтернативний варіант модулю виділення словосполучень із текстових даних із використанням платформи Apache Spark та клієнту для взаємодії з нею – PySpark.

По-перше, вхідні текстові дані розділяються на частини та зберігаються в RDD, який згодом перетворюється на DataFrame – розподілену структуру даних, яка за структурою є дуже схожою на таблицю реляційної бази даних.

На відміну від датафреймів, які обчислюються зразу в R і Python, обчислення датафреймів у Spark автоматично пришвидшується оптимізатором запитів. Оптимізатор Catalyst компілює операції, які були використані для побудови DataFrame, перед початком будь-яких обчислень у DataFrame. Оскільки

оптимізатор розуміє семантику операцій і структуру даних, він може приймати розумні рішення для прискорення обчислень.

Для Spark Dataframe існує два види оптимізації. По-перше, Catalyst застосовує логічні оптимізації запитів та операцій на даними, такі як виштовхування предикатів. По-друге, Catalyst компілює операції у фізичні плани для виконання та генерує байт-код JVM для цих планів, який часто є більш оптимізованим, ніж рукописний код. Фізичний план — це внутрішнє вдосконалення або оптимізація для Spark. Він створюється після оптимізованого логічного плану. Він також може виконувати оптимізацію нижнього рівня, наприклад, виключаючи дороге виділення об’єктів та зменшуючи виклики віртуальних функцій. У результаті можна очікувати покращення продуктивності програм Spark, якщо вони використовують DataFrame. Оскільки оптимізатор генерує байт-код JVM для виконання, користувачі PySpark та Python отримають таку ж високу продуктивність, як і користувачі Scala та Java.

Таким чином виконання завдань пов’язаних з роботою алгоритмів аналізу та оброблення текстових даних була перенесена на бік кластеру Spark, що значно прискорило роботу додатку.

3.5. Особливості реалізації API розроблюваної бази даних словосполучень

Сервер веб-додатку реалізований за допомогою Flask та має набір методів API, за допомогою яких можна зручно використовувати функціональні можливості системи.

Детальніший опис цих методів представлений у таблиці 3.1.

Таблиця 3.1 – Опис розроблених методів API

Метод	Путь	Деталі
GET	/upload	Отримання форми для завантаження текстових даних.
POST	/upload	Завантаження текстових даних для оброблення NLP пайплайном.
GET	/upload_spark	Отримання форми для завантаження текстових даних на оброблення за допомогою Spark.
POST	/upload_spark	Завантаження текстових даних для оброблення NLP пайплайном за допомогою Spark.
GET	/get_ngrams?word={word}	Отримання словосполучень, з якими вживається шукане слово.

За допомогою методу /upload система надсилає вхідні текстові дані на оброблення розробленими NLP алгоритмами, а потім вони попадають до сховища даних. Схожим чином працює метод API /upload_spark, але в даному випадку використовується альтернативна версія модулю виділення словосполучень з текстових даних, яка використовує Spark для оброблення текстів. Вхідні дані надсилаються файлом за допомогою запиту з типом вмісту multipart-form-data.

Метод /get_ngrams дозволяє отримати словосполучення, які вживаються разом зі словом, що передається як параметр запиту для цього методу API. У відповідь повертається список JSON об'єктів, які включають в себе знайдене словосполучення, розмітку за частинами мови та приклад використання даного

словосполучення в тексті. Запит до системи пошуку Elasticsearch, за допомогою якого відбувається пошук словосполучень за переданим до методу API словом, представлений на лістингу 4.

Лістинг 5. Запит до Elasticsearch для знаходження словосполучень за введеним словом

es.py

```
{
    "fuzzy": {
        "ngram": {
            "value": word,
        }
    }
}
```

Такий тип запиту називається fuzzy query – він повертає документи, які містять дані, схожі на пошукові, виміряні відстанню Левенштейна, яка означає кількість змін в один символ, необхідних для перетворення одного терміна в інший. [40]

Щоб знайти подібні текстові дані, fuzzy query створює набір усіх можливих варіантів або розширень пошукового терміну в межах заданої відстані редагування. Потім запит повертає точні збіги для кожного розширення. [41]

Висновки до третього розділу

В даному розділі було обґрунтовано використання певних інструментів, мови програмування та бібліотек для розроблення автоматичної генерації бази даних словосполучень української мови.

Для розроблення описаної системи була використана мова програмування Python, для розроблення API сервера використаний Flask, а для імплементації статистичних методів виділення словосполучень та нормалізації текстових даних – бібліотека NLTK. В якості сховища даних була використана система Elasticsearch, через те, що вона дозволяє швидко індексувати дані, є добре масштабованою та створена для того, щоб виконувати повнотекстовий пошук над даними.

Розглянуто також розроблені алгоритми кроків нормалізації текстових даних, імплементацію пошуку словосполучень, збереження даних до сховища та пошуку по них.

Також був описаний засіб прискорення виконання алгоритмів, що працюють з текстовими даними, за допомогою системи Apache Spark.

4. АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Результати роботи розробленої програмної системи

Для аналізу результатів роботи розробленої системи генерації бази даних словосполучень української мови в систему були завантажені текстові дані, вони були оброблені програмою та результати були збережені у сховищі даних. Розглянемо ці результати докладніше.

Вхідними текстовими даними для оброблення стала книга фентезійний роман Дж. Р. Р. Толкіна «Гобіт, або Мандрівка за Імлисті гори». Текст книги містить 80504 слова. [42]

Дані були оброблені системою та результатом став описаний у пункті 3.3 індекс Elasticsearch, наповнений знайденими словосполученнями. Всього в індексі налічується 22030 документів, а розмір індексу склав 14.2 МБ.

Для керування індексом Elasticsearch та аналізу отриманих даних була використана система Kibana. Розглянемо які дані збереглися в індексі після аналізу алгоритмами NLP. На Рисунку 22 можна побачити представлення документу з індексу в Kibana.

Field	Value
t _id	16f236996495bb2ae4cc05285f5f717609636dc46e6882c7932b4e5d
t _index	ngrams
# _score	0
t _type	_doc
t id	16f236996495bb2ae4cc05285f5f717609636dc46e6882c7932b4e5d
t ngram	дорогоцінний час
t pos	ADJF, NOUN
t sentence	Він знав, як мало в нього дорогоцінного часу, поки павуки отямляться і повернуться до дерев, де висіли гноми

Рисунок 22 – Документ, що містить інформацію про словосполучення «дорогоцінний час»

Бачимо, що в документі збережено саме знайдене словосполучення, розмітка за частиною мови та речення – приклад вживання.

Один з найкращих способів зрозуміти дані – це візуалізувати їх. Kibana дозволяє створювати різні чарти та графіки з даних, що збережені в індексах Elasticsearch. Тож було створено візуалізацію, яка показує розбиття всіх знайдених словосполучень за частинами мови. Її можна побачити на Рисунку 23.

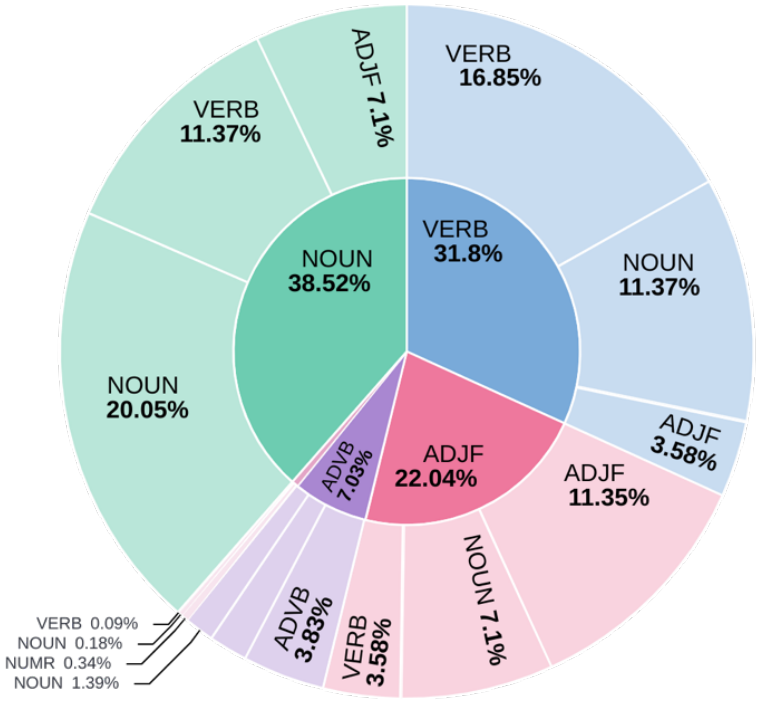


Рисунок 23 – Розбиття всіх знайдених результатів за шаблонами частин мови

На цій діаграмі можна побачити, що знайдена найбільша кількість словосполучень, що складаються з двох іменників, потім за кількістю йде *іменник* + *дієсло*во та *іменник* + *прикметник*.

Зібрані дані можна запросити за допомогою запиту на метод API /get_ngrams, тоді відповіддю на запит буде JSON об’єкт зі списком JSON представлень знайдених в Elasticsearch індексі документів. Приклад такого запиту можна побачити на Лістингу 6.

Лістинг 6. Приклад відповіді від методу API /get_ngrams

```
http://localhost:5000/get_ngrams?word=час

{
  "success": [
    {
      "ngram": "дорогоцінний час",
      "pos": ["ADJF","NOUN"],
      "sentence": "Він знав, як мало в нього дорогоцінного часу, поки павуки
отямляться і повернуться до дерев, де висіли гноми"
    },
    {
      "ngram": "час доводити",
      "pos": ["NOUN","VERB"],
      "sentence": "Час від часу їм доводилося обертатись і відбиватися від
потвор, що насідали ззаду, а декотрі павуки вже були вгорі над ними, на
деревах, кидаючи вниз свої довгі липучі мотузки"
    },
    ...
  ]
}
```

Загалом зібрані дані в індексі Elasticsearch та структура типу документу, в якій вони збережені дозволяє досить зручно досліджувати зібрані дані, або будувати на їх основі інші дослідження, наприклад, побудову систем текстових рекомендацій.

4.2. Аналіз ефективності роботи програми

Як було визначено в пункті 3.4 була розроблена альтернативна версія модулю виділення словосполучень з текстових даних, яка була прискорена за допомогою системи Apache Spark.

Apache Spark надає набір веб-інтерфейсів/інтерфейсів користувача для моніторингу стану програми, споживання ресурсів кластером та конфігурацій Spark. Користувацький інтерфейс допомагає зрозуміти, як Spark виконує завдання. Код програми — це набір інструкцій, які наказують драйверу виконати завдання Spark і дозволяють йому вирішувати, як саме цього досягти за допомогою

виконавців. Інструкції для драйвера називаються трансформаціями, а дія ініціює виконання. На Рисунку 24 можемо побачити, що Spark виконує чотири завдання для того, щоб обробити вхідні текстові дані:

- виконує дію map над RDD,
- витягує дані з RDD за допомогою операції collect,
- сериалізує дані для відправлення в Elasticsearch,
- завантажує дані до Elasticsearch.

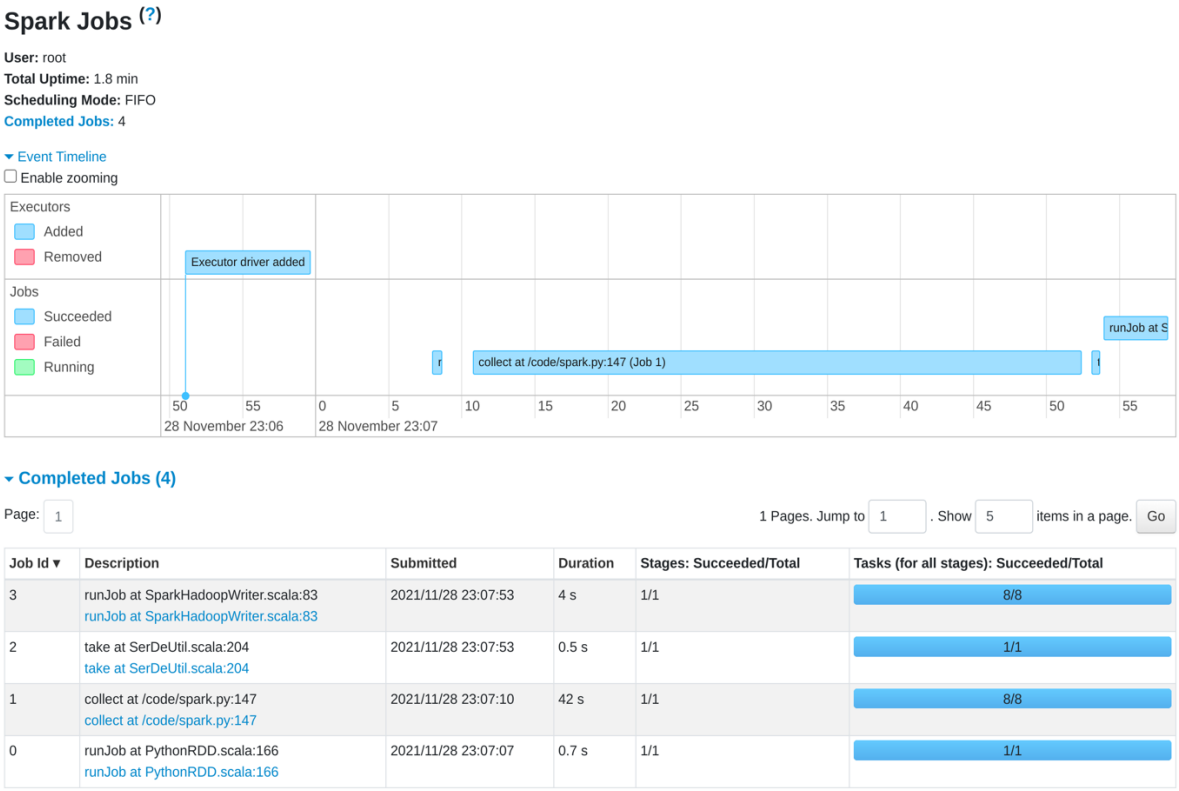


Рисунок 24 – Вкладка користувацького інтерфейсу «Spark Scheduling»

Більшість операцій над DataFrame або RDD відбуваються на стороні PySpark програми та не відправляються на драйвер. Та оскільки всередині вони використовують Scala код, ці операції все одно виконуються швидше, ніж просто за допомогою Python.

Apache Spark був використаний задля пришвидшення роботи програми, тому було виміряно наскільки пришвидшилося оброблення текстових даних у

порівнянні з простим використанням бібліотеки NLTK та виконані програми в одному Python процесі.

Для вимірювань ефективності був використаний інструмент під назвою Locust – простий у використанні інструмент для тестування продуктивності, для якого легко писати користувацькі програми використання. [43] Була написана невелика програма Python, яка визначає поведінку користувача системи, виконує запити до системи та записує час відповіді. На Рисунку 25 можна побачити результат виконання тесту.

Type	Name	Request Count	Failure Count	Median Response Time	Average Response Time	Min Response Time	Max Response Time
GET	/upload	1	0	9.248596001270926	9.248596001270926	9.248596001270926	9.248596001270926
POST	/upload	1	0	50981.42459099836	50981.42459099836	50981.42459099836	50981.42459099836
GET	/upload_spark	2	0	2.4696420005057007	2.497134501027176	2.4696420005057007	2.5246270015486516
POST	/upload_spark	1	0	17121.505516999605	17121.505516999605	17121.505516999605	17121.505516999605
	Aggregated	5	0	9	13623.434594600258	2.4696420005057007	50981.42459099836

Рисунок 25 – Звіт тестування інструментом Locust

В даній статистиці цікавими є дані за методами POST для /upload_spark та /upload, які відправляють текстові дані на оброблення для подальшого виділення словосполучень, з використанням Spark та ні відповідно. В колонках *Response Time* можна побачити, що час виконання зменшився для методу /upload_spark, у порівнянні з /upload, на 66,41%. Це є значним приростом у швидкості, а значить використання Spark є оправданим для збільшення ефективності програм, що оброблюють велику кількість текстових даних за допомогою NLP алгоритмів.

Висновки до четвертого розділу

В даному розділі були розглянуті результати роботи розробленої системи генерації бази даних словосполучень української мови. Був зроблений огляд даних, які потрапили в Elasticsearch після аналізу вхідних текстових даних. Розглянута структура документів, що зберігаються в індексі Elasticsearch та зроблено візуалізацію отриманих даних – чарт всіх словосполучень розбитих за частинами мови.

Також продемонстровано роботу API пошуку словосполучень за деяким словом, що користувач вводить у систему, розглянуто структуру відповіді, що повертається при запиті даного методу API.

Було показано, як працює Apache Spark у зв'язці з розробленою системою, розглянуто задачі, які він виконує про обробленні текстових даних.

Проведено тестування системи та визначено, що використання Apache Spark є оправданим з точки зору покращення ефективності та швидкодії оброблення текстових даних в рамках розроблених програмних засобів. Spark дійсно зменшив час виконання аж на 66,41%, що доводить необхідність його використання для більш ефективної роботи розробленого програмного забезпечення.

ВИСНОВКИ

В даній роботі було досліджено проблему формування бази даних словосполучень української мови за допомогою аналізу текстових даних.

Проаналізувавши основні методи знаходження колокацій у текстах, відомі наукові дослідження на дану тему та існуючі аналоги розроблюваної системи, був зроблений висновок, що не існує одного загальноприйнятого підходу, який б працював для будь-якого тексту будь-якою мовою. Проведений аналіз показав, високу варіативність підходів та важливість даного розділу NLP, а також те, що ця проблема є актуальною та викликає науковий інтерес.

В даному дослідженні було розглянуто проблему попереднього оброблення текстових даних і визначено набір кроків нормалізації, який підходить для досліджуваної проблеми, що повинно покращити результати роботи NLP алгоритмів. За допомогою проведеного експерименту доведено, що проведення нормалізації тексту, для роботи з алгоритмами NLP покращує ефективність та результати їх роботи.

Були розглянуті статистичні методи знаходження та ранжування кандидатів у словосполучення. Розглянуті наступні методи: простий підрахунок частот, t-критерій Стьюдента, критерій хі-квадрат, логарифмічна правдоподібність та метод взаємної інформації. За результатами проведеного експерименту визначено, що найефективніше для текстів саме українською мовою працює метод взаємної інформації. Також експериментальним шляхом визначено, що знайдені кандидати у словосполучення можна відфільтрувати за частинами мови, що покращить результати. Словосполучення, які залишаться після фільтрації стануть більш схожими на природну мову.

Як результат даного дослідження було розроблене програмне забезпечення – база даних словосполучень української мови, яка представляє собою бекенд систему, а в якості сховища даних використовує Elasticsearch. Були розроблені алгоритми кроків нормалізації текстових даних, імплементація пошуку словосполучень, збереження даних до сховища та пошуку по них. Також були описані засоби прискорення виконання алгоритмів NLP, за допомогою системи

Apache Spark. Проведення тестування системи показало, що використання Apache Spark є виправданим з точки зору покращення ефективності та швидкодії оброблення текстових даних в рамках розроблених програмних засобів. Spark зменшив час оброблення текстових даних на 66,41%, що доводить необхідність його використання для більш ефективної роботи розробленого програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Choueka, Yaacov. 1988. Looking for needles in a haystack or locating interesting collocational expressions in large textual databases. In Proceedings of the RIAO.
2. Firth, J. R. 1957. A synopsis of linguistic theory 1930-1955. In Studies in Linguistic Analysis, pp. l-32. Oxford: Philological Society. Reprinted in F. R. Palmer ted), Selected Papers of J. R. Firth 1952-l 959, London: Longman, 1968.
3. Michael Hoey. Lexical Priming: A New Theory of Words and Language. London: Routledge. 2005. ISBN 0-415-32863-2.
4. R. Mihalcea and T. Pedersen. 2003. An evaluation exercise for word alignment. In Rada Mihalcea and Ted Pedersen, editors, Proceedings of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts, pages 1–10.
5. Manning, Christopher & Schütze, Hinrich (1999). Foundations of Statistical Natural Language Processing. MIT Press. ISBN 0262133601.
6. Kenji Kita, Yasuhiko Kato, Takashi Omoto, and Yoneo Yano. 1994. A comparative study of automatic extraction of collocations from corpora: Mutual information vs. cost criteria. Journal of Natural Language Processing, 1(1):21–33.
7. Stefan Evert and Brigitte Krenn. 2001. Methods for the qualitative evaluation of lexical association measures. In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, Toulouse, France.
8. Snedecor, George Waddel, and William G. Cochran. 1989. Statistical methods. Ames: Iowa State University Press. 8th edition.
9. Church, Kenneth Ward, and Patrick Hanks. 1989. Word association norms, mutual information and lexicography. In ACL 27, pp. 76-83.
10. Dunning, Ted. 1993. Accurate methods for the statistics of surprise and coincidence. Computational Linguistics 19:61-74.
11. Smadja, Frank. 1993. Retrieving collocations from text: Xtract. Computational Linguistics 19:143-177.
12. Smadja, Frank A., and Kathleen R. McKeown. 1990. Automatically extracting and representing collocations for language generation. In ACL 28, pp. 252-259.

13. Goldman, J.-P. and Wehrli, E., “FipsCo: A Syntax-Based System for Terminology Extraction.” In Grammar and Natural Language Processing Conference, 2001.
14. Wu, C.-C. and Chang, J. S., “Bilingual Collocation Extraction Based on Syntactic and Statistical Analyses.” Computational Linguistics and Chinese Language Processing, vol. 9(1), pp. 1–20, 2004.
15. Vechtomova O., Robertson S.E., Jones S. (2003) Query expansion with long-span collocates. Information Retrieval, 6(2), pp. 251-273.
16. Thanopoulos, A., Fakotakis, N., and Kokkinakis, G., “Comparative Evaluation of Collocation Extraction Metrics.” In Proceedings of the LREC 2002 Conference, pp. 609–613, 2002.
17. Pearce, D., “A Comparative Evaluation of Collocation Extraction Techniques.” In Proc. of the 3rd International Conference on Language Resources and Evaluation (LREC 2002), Las Palmas, Spain, 2002.
18. da Silva, J. F. and Lopes, G. P., “A Local Maxima Method and a Fair Dispersion Normalization for Extracting Multi-Word Units from Corpora.” In 6th Meeting on the Mathematics of Language, Orlando, FL, pp. 369–381, 1999.
19. Sketch Engine [Електронний ресурс]: Create and search a text corpus | Sketch Engine. – Режим доступу: https://www.sketchengine.eu/.
20. WordSmith 4.0 [Електронний ресурс]: WordSmith. – Режим доступу: https://www.lexically.net/wordsmith/version4/.
21. Concordance software [Електронний ресурс]: Athelstan. – Режим доступу: https://www.athel.com/index.html.
22. Anagnostou, N.K. and Weir, G.R.S. (2006) Review of software applications for deriving collocations. In: ICT in the Analysis, Teaching and Learning of Languages, Preprints of the ICTATLL Workshop 2006, 2006-08-21 - 2006-08-22.
23. Ngram Statistics Package (NSP) [Електронний ресурс]: Ted Pedersen - Ngram Statistics Package / N-gram / Ngrams / Bigram / Bigrams. – Режим доступу: http://ngram.sourceforge.net/.
24. Shannon, C. E., “A Mathematical Theory of Communication.” The Bell System Technical Journal, vol. 27(1), pp. 379–423, 1948.

25. Hunspell [Електронний ресурс]: Hunspell: About. – Режим доступу: http://hunspell.github.io/.
26. UA-GEC [Електронний ресурс]: перший анотований GEC-корпус української мови вже у вільному доступі! – Режим доступу: https://ua-gec-dataset.grammarly.ai/.
27. Pecina, P., “An Extensive Empirical Study of Collocation Extraction Methods.” In Proceedings of the ACL Student Research Workshop, pp. 13–18, 2005.
28. Justeson, John S., and Slava M. Katz. 1995b. Technical terminology: some linguistic properties and an algorithm for identification in text. Natural Language Engineering 1:9-27.
29. Church, Kenneth W., and Robert L. Mercer. 1993. Introduction to the special issue on computational linguistics using large corpora. Computational Linguistics 19:1-24.
30. Pearson, Karl (1900). "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling" (PDF). Philosophical Magazine. Series 5. 50 (302): 157–175. doi:10.1080/14786440009463897.
31. Mood, Alexander M., Franklin A. Graybill, and Duane C. Boes. 1974. Introduction to the theory of statistics. New York: McGraw-Hill. 3rd edition.
32. NLTK [Електронний ресурс]: Natural Language Toolkit. – Режим доступу: https://www.nltk.org/.
33. Korobov M.: Morphological Analyzer and Generator for Russian and Ukrainian Languages // Analysis of Images, Social Networks and Texts, pp 320-332 (2015).
34. Apache Spark [Електронний ресурс]: Unified Engine for large-scale data analytics. – Режим доступу: https://spark.apache.org/.
35. Pandas [Електронний ресурс]: Python Data Analysis Library. – Режим доступу: https://pandas.pydata.org/.
36. Elasticsearch [Електронний ресурс]: Free and Open Search: The Creators of Elasticsearch, ELK & Kibana | Elastic. – Режим доступу: https://www.elastic.co/.
37. Kibana [Електронний ресурс]: Kibana: Explore, Visualize, Discover Data | Elastic. – Режим доступу: https://www.elastic.co/kibana/.

38. Flask [Електронний ресурс]: Welcome to Flask — Flask Documentation (2.0.x). – Режим доступу: https://flask.palletsprojects.com/en/2.0.x/.
39. Jinja [Електронний ресурс]: Jinja Documentation (3.0.x). – Режим доступу: https://jinja.palletsprojects.com/en/3.0.x/.
40. Levenshtein distance [Електронний ресурс]: Definition of Levenshtein distance, possibly with links to more information and implementations. – Режим доступу: https://xlinux.nist.gov/dads/HTML/Levenshtein.html.
41. Fuzzy query | Elasticsearch Guide [7.15] | Elastic [Електронний ресурс] – Режим доступу: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-fuzzy-query.html.
42. Толкін, Джон Роналд Руел = Гобіт, або мандрівка за Імлисті гори: повість-казка / Переклад з англійської О.М.Мокровольський; Обкладинка С. Железняк, ілюстрації С.Фесенко 348 с.: іл. ББК 84.4 ВЕЛ Т52. – ISBN 966-661-065-5.
43. What is Locust? – Locust 2.5.0 documentation [Електронний ресурс] – Режим доступу: http://docs.locust.io/en/stable/what-is-locust.html.

ДОДАТКИ

Додаток 1