Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів морфологічного аналізу слів української мови на основі машинного навчання
(тема)

Виконав:
Випускник 2 курсу, групи ІПЗм-21-1

Ковальов О.М.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

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

Керівник к.т.н. Бабій А.С.
(посада, прізвище)

Допускається до захисту
Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

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

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
« __ » _____ 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Ковальова Олександра Максимовича
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів морфологічного аналізу слів української мови на основі машинного навчання»

затверджена наказом університету від _____ №

2. Термін подання студентом роботи до екзаменаційної комісії «15» травня 2023 р.

3. Вихідні дані до роботи *електронні ресурси за вибраною тематикою, морфологічний аналіз слова, pymorphy2, python, jupyter notebook, Azure, Azure Databricks service*.

4. Перелік питань, що потрібно опрацювати в роботі *аналіз предметної галузі та постановка задачі, дослідження морфологічного аналізу за допомогою машинного*

навчання, опис прийнятих проектних рішень та технологій, розгортання середовища для проведення дослідження, проведення дослідження, аналіз результатів дослідження, висновок

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, ілюстрацій (слайдів) *мета завдання, обґрунтування доцільності розроблення, постановка задачі, методи і алгоритми, опис отриманих результатів, демонстраційні матеріали*

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	Бабій А.С.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	24.10.2022	*Виконано*
2	Огляд існуючих бібліотек для автоматичного аналізу тексту	03.11.2022	*Виконано*
3	Постановка задачі	14.11.2022	*Виконано*
4	Дослідження морфологічного аналізу за допомогою машинного навчання	16.02.2023	*Виконано*
5	Створення оточення та реалізація програмного застосунку	26.03.2023	*Виконано*
6	Підготовка пояснювальної записки	05.04.2023	*Виконано*
7	Перевірка роботи на антиплагіат	08.05.2023	*Виконано*
8	Нормоконтроль	09.05.2023	*Виконано*
9	Рецензування	10.05.2023	*Виконано*
10	Підготовка презентації та доповіді	11.05.2023	*Виконано*
11	Попередній захист	12.05.2023	*Виконано*
12	Занесення роботи в електронний архів	12.05.2023	*Виконано*
13	Допуск до захисту у зав. кафедри	15.05.2023	*Виконано*

Дата видачі завдання ____ _____ 2023 р.

Студент _____
(підпис)

Керівник роботи _____ к.т.н. Бабій А.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 69 с., 23 рис., 2 табл., 5 додаток, 11 джерел.

АНАЛІЗ, МАШИННЕ НАВЧАННЯ, МЕТОДИ, МОРФОЛОГІЯ, УКРАЇНСЬКА МОВА.

Об’єктом дослідження є аналіз та оцінка методів морфологічного аналізу слів української мови на основі машинного навчання.

Метою роботи є дослідження існуючих методів та бібліотек, що дозволяють провести морфологічний аналіз слів на основі машинного навчання, з ціллю виявлення переваг та недоліків, задля виявлення найкращих інструментів аналізу та внесення пропозицій для їх покращення.

Після проведення аналізу предметної галузі було сформовано подальші кроки для проведення дослідження.

У ході роботи було проаналізовано методи автоматичного аналізу українських слів з використанням машинного навчання, розроблено програмний застосунок, а також була розроблена відповідна документація.

ANALYSIS, MACHINE LEARNING, METHODS, MORPHOLOGY, UKRAINIAN LANGUAGE.

The object of the study is the analysis and evaluation of methods of morphological analysis of Ukrainian words based on machine learning.

The purpose of the work is to research existing methods and libraries that allow morphological analysis of words based on machine learning, with the aim of identifying advantages and disadvantages, in order to identify the best analysis tools and make suggestions for their improvement.

After conducting the analysis of the subject area, further steps for conducting the research were formed.

In the course of the work, methods of automatic analysis of Ukrainian words using machine learning were analyzed, a software application was developed, and relevant documentation was developed.

Я, *Ковальов Олександр Максимович,* студент групи ІПЗм-21-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів морфологічного аналізу слів української мови на основі машинного навчання», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(а) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

Вступ 9
1 Аналіз предметної галузі та постановка задачі 10
1.1 Морфологічний розбір слова 10
1.2 Автоматичний аналіз і синтез тексту 11
1.3 Інструменти морфологічного аналізу 14
1.4 Постановка завдань дослідження 19
2 Дослідження морфологічного аналізу за допомогою машинного навчання 20
2.1 Використання машинного навчання для морфологічного аналізу 20
2.2 Вивчення морфологічних правил 22
2.3 Контроль процедури аналізу 24
2.3.1 Неоднозначність 24
2.3.2 Тип та порядок 26
3 Опис прийнятих проектних рішень та технологій 30
3.1 Опис прийнятих проектних рішень 30
3.2 Внутрішня будова бібліотеки pymorphy2 32
3.2.1 Словники 32
3.2.2 Розбір не словникових слів 36
4 Опис програмної реалізації 40
4.1 Створення оточення для програмної реалізації 40
4.2 Реалізація програмного застосунку 44
5 Аналіз проведеного дослідження 50
Висновки 52
Перелік джерел посилання 53
Додаток А Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії 55
Додаток Б Звіт результатів перевірки кваліфікаційної роботи на унікальність тексту 56

Додаток В Слайди презентаці 57
Додаток Г Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015 69

ВСТУП

Автоматичний морфологічний аналіз (АМА) [1] – це однин із етапів роботи систем автоматичного аналізу тексту. У результаті роботи АМА кожному слововживанню приписуються значення граматичних категорій (частина мови, рід, число, відмінок, час, вид, тощо).

У наш час існує безліч онлайн ресурсів для проведення морфологічного аналізу слів, деякі з яких використовують й машинне навчання для проведення такого аналізу. Але важливо розуміти, які методи та бібліотеки справляються зі своїм завданням краще, та які мають переваги та недоліки.

Тож метою кваліфікаційної роботи є аналіз предметної галузі, а саме дослідження існуючих методів та бібліотек, що дозволяють провести морфологічний аналіз слів на основі машинного навчання. Це надасть можливість виділити основу для майбутнього дослідження.

Далі буде розглянуто алгоритми за темою кваліфікаційної роботи та внутрішню будову роботи обраної бібліотеки, та продемонстровано її функціонал. Для виконання практичної частини роботи буде сформовано датасет для перевірки продуктивності роботи бібліотеки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Морфологічний розбір слова

Морфологічний розбір слова [2] – це характеристика слова як частини мови. Він включає опис значення слова, його граматичних особливостей, а також ролі, яку дане слово грає в реченні.

Морфологічний аналіз різних частин мови можна звести до виконання наступних кроків [2]:

- частина мови, загальне значення;
- морфологічні ознаки;
- початкова форма слова;
- постійні ознаки;
- непостійні ознаки;
- синтаксична роль у реченні.

Однак кожна з частин мови має властиві лише їй постійні та непостійні граматичні ознаки.

План морфологічного розбору слова такий:

- вказати частину мови та її значення, на яке питання відповідає слово;
- поставити слово у початкову форму: Ім.п., од.ч. – для іменників, Ім.п., од.ч., м.р. – для прикметників, невизначена форма – для дієслів (що зробити?);
- визначити постійні ознаки: загальне або власне, живе чи не живе, рід і відмінювання у іменників; вид, повернення, перехідність і відмінювання у дієслова; розряд за значенням, ступінь порівняння, повна або коротка форма прикметників;
- охарактеризувати форму, в якій слово вжито: у іменників визначити число і відмінок, у прикметників – ступінь порівняння, коротку або повну форму, число, відмінок і рід; у дієслів – спосіб, час, число, рід або особа, якщо є;
- роль у реченні – показати, яким членом є слово у реченні: другорядним чи головним. Іноді потрібно виписати словосполучення та показати його синтаксичну роль графічно.

Тобто у морфологічному розборі різних частин мови є суттєві відмінності.

1.2 Автоматичний аналіз і синтез тексту

Автоматизований аналіз і синтез тексту є важливими завданнями в комп'ютерній лінгвістиці як з точки зору розробки лінгвістичних основ для створення штучного інтелекту, так і з точки зору задоволення практичних потреб людини, наприклад створення ефективних систем машинного перекладу.

Автоматична обробка текстів має широке практичне застосування: створення словників та інтелектуальних пошукових систем, розробка лінгвістичних процесорів для забезпечення спілкування з користувачами природною мовою, класифікація емоційного забарвлення тексту[3], визначення значення текстових документів[4], автоматична абстракція тексту та ін.

При обробці текстів природною мовою необхідно ідентифікувати певні текстові елементи. Вхідними даними у процес є текст природною мовою, а вихідними – певні структури даних, що допускають подальшу автоматичну або ручну обробку тексту. Морфологічний аналіз є важливою частиною процесу попередньої обробки тексту.

Морфологія – розділ граматики, що вивчає граматичні властивості слів[5]. Граматичними властивостями слів є граматичні значення, способи вираження граматичних значень. Морфологія вивчає будову частин мови. Основна мета – розбити словоформу на дві частини.

Морфологічна розмітка під час автоматичної обробки текстів природною мовою є основою як для морфологічного аналізу, так і для інших форм аналізу – синтаксичного та семантичного.

Труднощі морфологічного аналізу випливають з однієї з властивостей будь-якої мови – її змінюваності. Умовно весь лексичний склад мови можна поділити на дві частини. Перший – це певний набір слів (лексика), що виник у процесі розвитку мови. Він має достатню стійкість і є основою для створення текстів писемного й

усного мовлення.

Другий лексичний пласт характеризується більшою рухливістю. Він включає нові слова, які існують в мові порівняно недавно, наприклад, імена власні і словотвірні, варіанти вже відомих слів тощо. Тому важливо мати алгоритми, вбудовані в аналізатор і розраховані на обробку як відомих, так і нових слів.

В українській мові багато словозмін, але запам'ятовування всіх можливих словоформ потребує значних ресурсів пам'яті. Наприклад, українська іменник, що змінюється за кількістю (єдиним і множиною) і відмінком (7 відмінків), має 14 словоформ. А у дієслова, що змінюється і має наказовий спосіб залежно від особи (1-ї, 2-ї та 3-ї особи), часу (теперішнього, майбутнього і минулого часу), числа (єдиного та множини) і прислівника, відповідно, ще одна більша кількість словесних форм.

Наразі існує кілька морфемних словників слів, але майже всі вони є паперовими або цифровими PDF-файлами, що робить їх неможливими для використання в автоматизованій системі обробки текстів.

На сьогодні основними підходами до розв'язання проблеми морфологічного аналізу є наступні:

- морфологічний аналіз на основі словників;
- морфологічний аналіз без використання словників.

Якщо використовувати аналіз із застосуванням словників, тоді:

- словники дадуть максимальну інформацію за формою відомого слова;
- на реальних текстах будуть збої через наявність помилок;
- виникне проблема повноти словників (всі можливі імена, прізвища, нові слова тощо).

Методи нормалізації слів без використання словника використовують алгоритми, призначені для перетворення слів у різні граматичні форми. Їх можна поділити на:

- ймовірнісно-статичні методи;
- лексикону основ і суфіксів.

Методи першої групи вимагають використання великої вибірки, а для другої

групи – великої кількості лексиконів і способів їх отримання.

Ми розглянемо деякі існуючі алгоритми морфологічного розбору слів.

У першому алгоритмі аналіз морфологічної структури слова складається на основі асоціативно-статистичного підходу, заснованого на природному накопиченні асоціацій між образами і закріпленні рефлексів у вигляді повторення. Перевага пропонованого підходу перед іншими методами вирішення поставленого завдання, такими як синтаксично-семантичний підхід, онтології, запити до бази даних, полягає в тому, що не потрібна попередня робота з висококваліфікованими фахівцями. Вихідною інформацією на вирішення цього завдання є словник мовних образів.

У наступному прикладі використовується Porter Stemming алгоритм, у якому морфологічний аналіз текстового вмісту полягає в пошуку основи слів. Набір правил послідовно застосовуються, щоб залишити лише основу слова, спосіб вирізання суфіксів, префіксів тощо. Ключові частини слова вирізаються простою функцією, яка вибирає слова. Далі в таблицю записується основа кожного слова. Але недоліком цього способу є те, що необхідно враховувати всі правила словотворення в українській мові (флексії залежно від роду, відмінювання, суфіксів, префіксів, а також з боку частини мови, однини й множини тощо). Наприклад, цей алгоритм не працює з кількома словами. У міру збільшення правил обробки навантаження збільшується у геометричній прогресії. Наприклад, стоїть завдання перевірити та визначити ключові частини слова за 101 статтею на день, кожне слово має бути перевірено аналізатором приставок, закінчень та суфіксів. І тут складність такого алгоритму зростає до критичної межі.

Інший алгоритм шукає таблиці та скорочує закінчення та суфікси. Після нормалізації (лематизації) такі таблиці повинні містити всі можливі варіанти всіх слів і словоформ. Перевагою цього підходу є обробка всіх винятків із правил, а також простота та швидкість. Недоліком є те, що таблиці повинні містити всі словоформи всіх слів, а це означає, що цей алгоритм не працює зі словами, які недавно з’явилися в мові. При цьому таблиці слів і словоформ можуть стати дуже великими. Розміри таблиць пошуку невеликі лише для мов із не складною

морфологією, таких як англійська. Однак в інших мовах кількість варіантів словоформи з однією основою може досягати навіть сотень, наприклад, турецькою мовою. Слова скорочуються відповідно до правил, на яких заснований цей алгоритм.

У цьому алгоритмі при усіченні суфіксів і закінчень кількість правил нормалізації словоформ менше, ніж таблиця всіх слів і словоформ, тому ця частина алгоритму досить компактна і продуктивна. Але цей алгоритм іноді може помилятися та робити неправильні висновки та спотворювати форму слова.

Також широко поширений метод, заснований на системі машинного навчання. Система машинного навчання аналізує заданий текст і навчається на ньому, розпізнає певні шаблони та робить деякі узагальнення на основі цих шаблонів. Використовуючи отриману інформацію про властивості словоформ, які регулярно зустрічаються в усіх контекстах, що пройшли аналіз, можна робити прогнози щодо найбільш вірогідного граматичного тлумачення словоформи в нових текстах.

Методи машинного навчання з учителем після навчання роблять імовірнісні прогнози на корпусі текстів, повністю або частково розмічених інформацією про словоформи, а методи машинного навчання дозволяють працювати з нерозміченим корпусом. Системи морфологічного аналізу, засновані на цьому методі, були створені для англійської мови та адаптовані до деяких флективних мов. Ці аналізатори використовують техніку правил перетворення, одержаних у результаті машинного навчання. У цілому нині точність аналізу корпусів англомовних текстів імовірнісними аналізаторами перевищує 97%.

1.3 Інструменти морфологічного аналізу

У наш час існує вже деяка кількість інструментів та бібліотек, що вирішують проблему автоматичного морфологічного аналізу слів на основі машинного навчання. Деякі з них є більш поширеними, деякі менш популярними, деякі

бібліотеки підтримують аналіз слів української мови, а деякі ні.

Одним з найбільш поширених аналізаторів в українському Natural Language processing й не тільки є pymorphy2 [6]. Його код є відкритим й доступним для використання після завантаження з GitHub репозиторію (див. рис. 1.1).

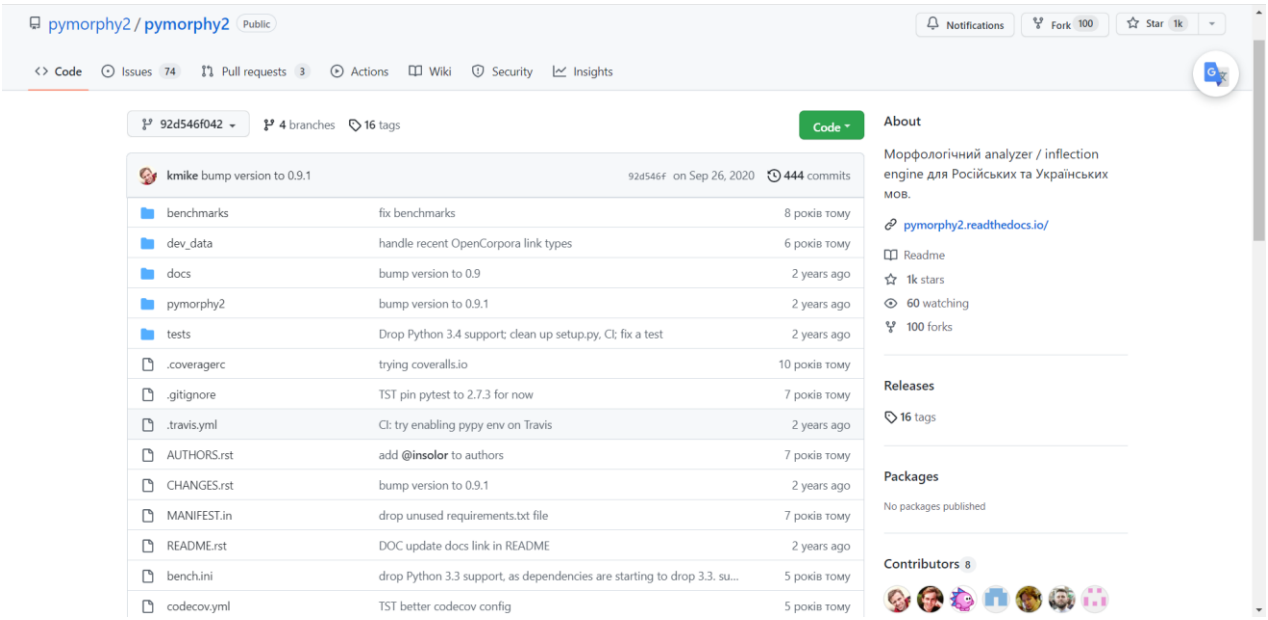

Рисунок 1.1 – Бібліотека pymorphy GitHub репозиторій

Під час роботи даний аналізатор використовує словник OpenCorpora. Для не знайомих слів будуть збудовані гіпотези на основі машинного навчання, що дає можливість аналізу слів, не присутніх у словнику. Сама бібліотека працює досить швидко та може аналізувати від кількох тисяч слів за секунду, до сотні тисяч слів за секунду. Продуктивність роботи аналізатора залежить від виконуваної операції, обраного інтерпретатора та встановлених пакетів. Споживання оперативної пам’яті бібліотекою буде в районі від 10 до 20 мегабайтів. Бібліотека підтримує морфологічний аналіз слів української та російської мови.

Морфологічний аналізатор pymorphy2 має змогу виконувати наступний перелік функцій:

– приводити слова до нормальної форми. Наприклад, “люди” до “людина”, або “гуляв” до “гуляти”;

− ставити слова у потрібну форму. Наприклад, ставити слово у множину, змінювати відмінок слів і тому подібне;

− повертати граматичну інформацію про слово. Наприклад число, рід, відмінок, частину мови тощо.

Наступним інструментом для морфологічного аналізу слів, що було розглянуто, є консольне застосування MyStem. Ця програма дає можливість виконувати морфологічний аналіз тексту та вміє будувати гіпотетичні розбори для більшості слів, включаючи ті слова, яких не має у словарі.

Словник представлений у вигляді набору спроб [cormen1990] перевернутих коренів із всеможливими суфіксами. Кордон між коренем і суфіксом береться або з вхідного файлу, якщо він явно представлений або обчислений автоматично, як довжина загальної частини усіх словоформ у парадигмі. Далі представлений опис алгоритму:

− перехід у слово з правого кінця за допомогою спроби всіх можливих суфіксів. В результаті ми будемо мати всі можливі позиції основи/суфікса кордону за один прохід;

− виконання наступних кроків починаючи із найглибшої основи;

− використання двох останніх літер можливої основи, як індексів для отримання відповідної основи спроби. Якщо основа не має двох останніх літер, то буде виконано перехід до наступної межі основи/суфікса;

− осмислення відповідної спроби основи й спроба знайти кінцевий лист на краю слова, зібравши всі точки розгалуження в одну перепустку;

− словникове слово. Якщо знайдено кінцевий лист на краю слова, то йде порівняння в відповідну флексійну модель (її ідентифікатор зберігається в основі спроби, як спеціальна літера після першої букви основи) із суфіксом слова. Якщо це збігається, це означає, що ми знайшли словникове слово;

− невідоме слово. Якщо немає термінала на краю слова або флексійна модель не збігається з суфіксом, тоді виконуються наступні кроки;

− пройдіть всі раніше зібрані точки розгалуження основи, намагаючись починатися з найглибшої, і знайдіть усі основи-кандидати, які можуть бути

моделями для даного невідомого слово. Перевірте їхні флексійні моделі з поданим суфіксом.

Алгоритм підтримує 2 додаткових режими:

- перевіряти тільки словникові слова;
- знайти тільки одного кандидата (це корисно для відповідності виведення один до одного)

З недоліків аналізатора можна виділити те, що він не підтримую аналіз слів української мови.

Наступним засобом морфологічного аналізу слів, який ми розглянемо, буде бібліотека phpmorphy [7]. Код бібліотеки є відкритим й доступним для використання після завантаження з GitHub репозиторію (див. рис. 1.2)

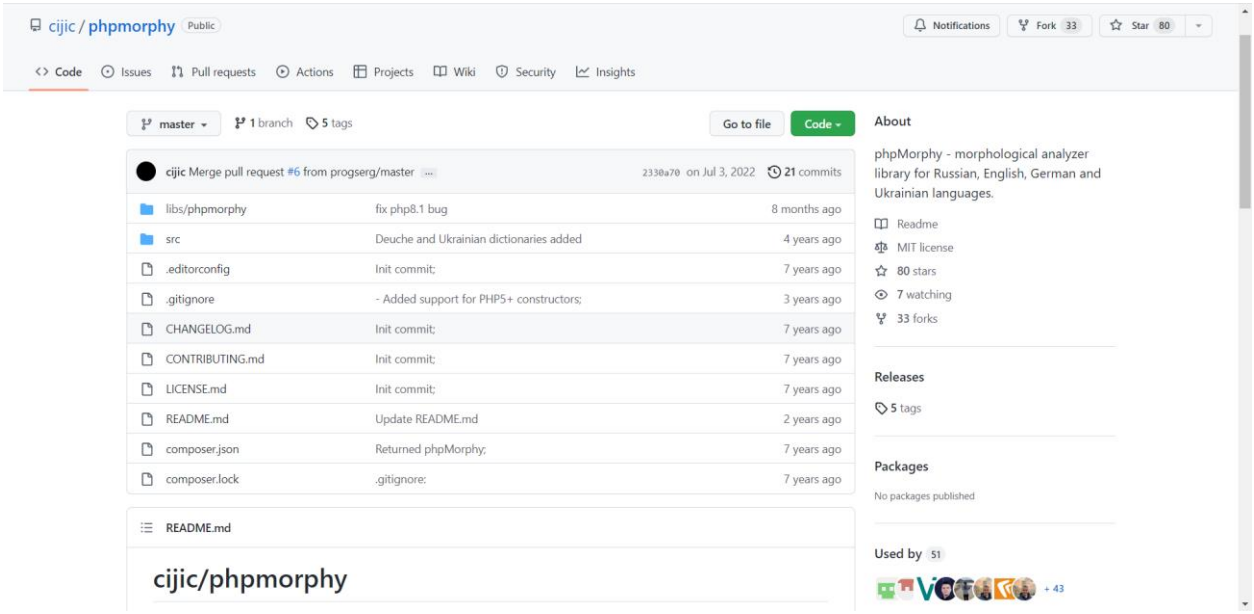

Рисунок 1.2 – Бібліотека phpmorphy GitHub репозиторій

Дана бібліотека реалізована для платформи php й використовується для морфологічного аналізу слів української, англійської, російської та німецької мов. cijic/phpMorphy – це оболонка Laravel для бібліотеки phpMorphy з підтримкою PHP7.

Морфологічний аналізатор phpMorphy дозволяє вирішувати наступний перелік завдань:

- лематизація (отримання нормальної форми слова);
- отримання усіх форм слова;
- отримання граматичної інформації про слово (частина мови, відмінок, відмінювання тощо);
- зміна форми слова відповідно до заданих граматичних характеристик;
- зміна форми слова за заданим зразком.

Бібліотека phpMorphy вміє працювати з наступним переліком мов:

- українська;
- англійська;
- російська;
- німецька (AOT);
- естонський (на основі ispell).

У морфологічному аналізаторі є можливість додати підтримку інших мов за допомогою myspell словника.

Бібліотека підтримує різноманітні кодування:

- всі однобайтові (windows-1251, iso-8859-* тощо);
- unicode кодування – utf-8, utf-16le/be, utf-32, ucs2, ucs4.

Останні два інструменти для морфологічного аналізу слів, що будуть розглянуті, це TreeTagger [8] та FreeLing [9].

TreeTagger – це інструмент для анотування тексту частиною мови та інформацією про леми. Він був розроблений Гельмутом Шмідом у рамках проєкту ТС в Інституті комп’ютерної лінгвістики Університету Штутгарта. TreeTagger успішно використовувався для позначення тегами німецькою, англійською, французькою, італійською, датською, шведською, норвезькою, голландською, іспанською, болгарською, російською, португальською, галісійською, грецькою, китайською, суахілі, словацькою, словенською, латинською, естонською, польською, Тексти перською, румунською, чеською, коптською та старофранцузькою мовами та адаптовані до інших мов, якщо доступні лексикон і тегований вручну навчальний корпус.

Проєкт FreeLing був створений і наразі очолюваний Луїсом Падро як засіб

надання спільноті результатів дослідження, проведеного дослідницькою групою обробки природної мови UPC.

FreeLing – це бібліотека C++, яка надає функції мовного аналізу (морфологічний аналіз, виявлення іменованих об’єктів, PoS-теги, аналіз, усунення неоднозначності слів, маркування семантичних ролей тощо) для різних мов (англійська, іспанська, португальська, італійська, французька, німецька, російська, каталонська, галісійська, хорватська, словенська та інші).

FreeLing також надає інтерфейс командного рядка, який можна використовувати для аналізу текстів і отримання вихідних даних у потрібному форматі (XML, JSON, CoNLL).

1.4 Постановка завдань дослідження

На основі аналізу предметної області, було виявлені наступні задачі для спланованого дослідження:

- дослідження алгоритмів морфологічного аналізу за допомогою машинного навчання;
- підготовка рішень, що будуть використані у проекті
- аналіз можливостей бібліотеки pymorphy2 для морфологічного аналізу тексту
- створення та підготовка датасету для перевірки можливостей автоматичного морфологічного аналізатора pymorphy2;
- підготовка тестового застосунку на мові python для морфологічного аналізу створеного датасету;
- аналіз результатів виконаного дослідження;
- визначення переваг та недоліків інструменту pymorphy, на основі виконаного дослідження, аналіз роботи pymorphy.

В наступному розділі буде проведено дослідження морфологічного аналізу за допомогою машинного навчання.

2 ДОСЛІДЖЕННЯ МОРФОЛОГІЧНОГО АНАЛІЗУ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

2.1 Використання машинного навчання для морфологічного аналізу

Автоматичний морфологічний аналіз (АМА) все ще є широко обговорюваною темою в обробці природної мови. Метою СМА є розуміння внутрішнього механізму утворення словоформ у мові. Морфологічний аналізатор може надати цінну інформацію для інших комп'ютерних лінгвістичних завдань, таких як лематизація, синтаксичний аналіз, машинний переклад, пошук інформації, кластеризація тексту та багато інших.

Голдсміт[10] класифікує роботу в автоматичному морфологічному аналізі на чотири категорії. Класифікація виконана з наголосом на тому, як отримані морфологічні правила. Перша категорія визначає межі морфем на основі ступеня передбачуваності n+1-ї літери з урахуванням перших n літер. Другий використовує граматику n-грам і високу ймовірність отримати внутрішню структуру морфеми. Третя категорія намагається виявити правила через фонологічні зв'язки між парами споріднених слів. Четвертий прагне до аналізу мови, який є найбільш стислим і, отже, зводиться до набору правил для мови.

Два фактори важливі для досягнення точного автоматичного морфологічного аналізу. Одним фактором є побудова набору морфологічних правил, а іншим є процедура морфологічного аналізу. Відсутність або недостатня продуктивність будь-якого з них погіршує загальну здатність морфологічного аналізатора. Таким чином, при побудові алгоритму для завдання необхідно враховувати обидва фактори.

Машинне навчання є перспективною альтернативою отримання морфологічних правил. Цей метод може уникнути таких проблем, як дорога людська праця, непослідовність правил, і може надати додаткову статистичну інформацію, яку можна використовувати в процедурі морфологічного аналізу. На основі типу інформації, яка використовується в задачі машинного навчання, ми

можемо отримати два класи: контрольоване навчання та неконтрольоване навчання. Клас контрольованого навчання використовує лексичну базу даних з морфологічною інформацією. Хорошим прикладом такої лексичної бази даних є CELEX2 [11]. У класі неконтрольованого навчання використовується лише список слів з інформацією про частоту або без неї.

Що стосується процедури морфологічного аналізу, то існує дві популярні методики. Одним із методів є стемінг, представлений Портером. Алгоритм, поданий Портером, складається з двох етапів. Перший етап десуфіксації, який віднімає попередньо визначені закінчення зі слів, і крок перекодування, який додає можливу кінцеву частину до рядка, отриманого на попередньому етапі. Ці два етапи можна виконувати послідовно або одночасно. Одна відмінна риса алгоритму полягає в тому, що він не використовує словник, що робить його дуже компетентним для аналізу.

Іншою технікою є морфологічний аналіз, представлений моделлю дворівневої морфології, запропонованою Коскеннімі. Ця модель розглядає морфологічний аналіз і морфологічну трансформацію в різних фонологічних ситуаціях і кодує відповідність між поверхневою формою та лексичною формою за допомогою перетворювача кінцевого стану. Наприклад:

– лексична форма: s p e c i f y + s;

– поверхнева форма: s p e c i f i e s.

Першою системою морфологічного аналізу, яка приймає цю модель, є KIMMO [12]. Система має дві частини: правила та лексичну інформацію, таку як морфемна структура та морфосинтаксичне обмеження.

Морфологічний аналізатор, проілюстрований далі, належить до першого класу класифікації Голда. Система націлена на високу точність морфологічного аналізу англійської мови з морфологічними правилами, отриманими шляхом неконтрольованого машинного навчання. Аналізатор застосовує літерну ймовірність переходу, запропоновану Кешавою та Пілтером[13] для вивчення морфологічних правил, а також для усунення неоднозначності морфологічного аналізу. Початкова оцінка аналізатора показує результат із точністю 88,42%,

запам’ятовуванням 78,46 і F-оцінка 83,14%, що перевершує найкращі результати з англійської мови, зазначені в Unsupervised Segmentation of Words into Morphemes – Challenge 2005.

2.2 Вивчення морфологічних правил

Запропонований Кешавою та Пілтером підхід було перевірено для вивчення правил афіксів зі списку слів, використовуючи список слів різних масштабів. Експерименти показують, що правила афіксів, отримані зі списком слів різних масштабів, відрізняються у великому діапазоні. Різниця в охопленні та правильність отриманих правил також призводить до різної продуктивності системи. Для вивчення правил афіксів будується одне пряме лексикографічне дерево та одне зворотне лексикографічне дерево. Потенційні афікси розпізнаються за допомогою процедури підрахунку балів. Процедура оцінювання складається з двох етапів. На першому кроці розглядається словоформа αABβ. Якщо наступні три умови:

– αA може бути знайдена у словнику;

– P(A|α) ≈ 1 , тобто в прямому лексикографічному дереві ймовірність переходу від α до A наближається до 1;

– P(B|αA) < 1 , тобто в прямому лексикографічному дереві ймовірність переходу від αA до B менше 1.

Bβ вважається кандидатом на суфікс. Афікс також можна оцінити за допомогою зворотного лексикографічного дерева з симетричними умовами. Другий крок оцінює Bβ шляхом перевірки всіх словоформ, що закінчуються рядком. Функція підрахунку балів наведена нижче:

$$score(B\beta) = \{(a) += AwardScore\,(b) -= PenalryScore \qquad (2.1)$$

Варіант (a) вказує, що якщо рядок, який закінчується на Bβ, відповідає всім

умовам, згаданим у першому кроці, оцінка Bβ як суфікса зростає на AwardScore. Варіант (b) вказує, що якщо рядок, який закінчується на Bβ, не відповідає всім умовам, оцінка Bβ як суфікса зменшується на PenaltyScore. Після перевірки всіх рядків, які закінчуються на Bβ, застосовується золотий стандарт. Якщо socre(Bβ) більше 0, рядок Bβ вважається суфіксом. В іншому випадку від нього відмовляються. Префікси обробляються подібним чином із зворотним деревом.

У таблиці 2.1 наведено результати експерименту, отримані з різними словниковими шкалами.

Таблиця 2.1 – Результати експерименту перевірки підходу Кешави та Пілтера

	Словниковий номер слоформи	Номер префіксу	Номер суфіксу	Точність	Відкликання	F-оцінка
1	167377	683	1322	87.52%	77.14%	82.00%
2	57046	694	989	88.42%	78.46%	83.14%
3	14760	373	1584	90.57%	72.43%	80.49%

Для прямого та зворотного лексикографічних дерев було використано корпус із 24 447 034 лексем. Як показано в списку, шкала словника, яка використовується для перевірки умови «a» на першому кроці, є головним фактором, що впливає на кількість правил префіксів і суфіксів, а також на продуктивність. Причину, яка може пояснити цю різницю, знайти неважко. В англійській мові не всі словоформи можна використовувати як основу, з якої можна утворити похідні слова. Наприклад, власне ім’я, яке містить такі власні імена, як «Ann», «Al», призводить до неправильної оцінки алгоритму.

2.3 Контроль процедури аналізу

Широко охоплюючий і правильний набір правил афіксів є необхідною

умовою для точного морфологічного аналізу. Але саме по собі це не гарантує успішного аналізу. Процедура, за якою проводиться аналіз, також має вирішальне значення. Далі розглядаються два важливі аспекти з точки зору контролю процедури аналізу. Одне – усунення неоднозначності. Інший порядок правил афіксів.

2.3.1 Неоднозначність

У морфологічному аналізі існують неоднозначності. Добре розуміння типів неоднозначностей, безсумнівно, допомагає вирішити неоднозначності. Класифікація неоднозначності в китайській сегментації також може застосовуватися тут. Таким чином, ми маємо два типи неоднозначності в морфологічному аналізі: перехресна неоднозначність і комбінаторна неоднозначність.

Перехресна неоднозначність означає тип рядків, які мають більше одного можливого перехресного аналізу. Наприклад, заданий рядок ABCD, B, CD, C і D є потенційними афіксами. Таким чином, ключем до перехресної неоднозначності є визначення місця морфологічної межі. Розглянемо наступний приклад:

- anthropophagous → anthropophagous s
  anthropophagous → anthropophag ous;
- beneficence → beneficen ce
  beneficence → benefi cence;
- fieldmice → fieldmi ce
  fieldmice → field mice.

Перехресна неоднозначність є явищем, яке часто зустрічається в морфологічному аналізі, і є важливим фактором, який спричиняє недостатню продуктивність аналізатора.

Кешава та Пілтер пропонують розв’язати перехресну неоднозначність за допомогою ймовірностей переходу між літерами. Розглянемо наступні приклади:

- action → acti on TransProb(i,o) = 0.583511
  action → act ion TransProb(t,i) = 0.500998;
- aeroplaces → aeroplane s TransProb(e,s) =0.295008
  aeroplaces → aeroplan es TransProb(n,e)=0.996983.

Справжні морфологічні межі в другому прикладі зазвичай мають меншу ймовірність переходу “i”, таким чином, правильно визначені. У даному експерименті фільтр встановлювався рівним 0,40, а словоформа з імовірністю переходу менше 0,40 розділялася.

Але цей метод не працює для слів, які не видно в вибірці, навченому для ймовірності переходу літер. У наступному прикладі, незважаючи на те, що вибірка велика, деякі слова все ще не зустрічаються:

- pilation → pilati on TransProb(i,o)=0;
- pilation → pilat ion TransProb(t,i)=0.

Імовірність переходу не може допомогти в усуненні неоднозначності в таких випадках. У нашій системі ми вдалися до наближення, яке бере найбільш часто використовуваний афікс як результат аналізу. Таким чином, у третьому прикладі результатом є «ion», який, як виявилося, правильний.

Комбінаторна неоднозначність у морфологічному аналізі стосується того факту, що аналізатор не може вирішити, чи має словоформа мати афікс чи ні. Тобто, маючи словоформу AB, у якій B є потенційним суфіксом, ми маємо вирішити, чи AB є окремою словоформою, а B не є афіксом, чи B є справжнім афіксом, а A є основою. Таким чином, ключ до комбінаторної неоднозначності полягає в тому, щоб вирішити, чи існує морфологічна межа всередині словоформи. Ось кілька прикладів:

- analects → analect s
  analects → analects;
- potion → pot ion
  potion → potion;
- thrive → thr ive
  thrive → thrive.

Комбінаторну неоднозначність розв’язати складніше, ніж перехресну неоднозначність. Простий кінцевий автомат точно не може вирішити проблему, оскільки кожне правило може мати виняток.

Щоб вирішити комбінаторну неоднозначність, також було вирішено покладатися на ймовірність переходу букв. Рішення можна пояснити, спостерігаючи наступне:

- letter → lett er TransProb(t,e) = 0.944217;
- alexic → alex ic TransProb(x,i) = 0.0216901;
- consumer → consum er TransProb(m,i) = 0.516369;
- encode → en code TransProb(c,n) = 0.

Морфологічна межа зазвичай має набагато меншу ймовірність переходу. Також було встановилено фільтр рівним 0,4 і розділено ті, які мають перехідні ймовірності, менші за фільтр. як, наприклад, четверте слово у прикладі вище, де ймовірність переходу літери дорівнює 0 і, отже, нижча за фільтр, також вважається справжнім афіксом, якщо рядок, що залишився після віднімання, є словом у словнику. У таких випадках перехідна ймовірність, що дорівнює 0, вказує на те, що, хоча слово не видно в словнику, воно безумовно містить морфему та словоформу, а отже, ймовірно, нове слово.

Експеримент показує, що згаданий вище метод значною мірою покращує продуктивність. Однак він не може вирішити всі комбінаторні неоднозначності в аналізі. Так само, як комбінаторна неоднозначність у китайській сегментації, проста перехідна ймовірність не може вирішити такі специфічні проблеми. Для усунення неоднозначності цього типу потрібна багатша контекстуальна інформація, така як граматична категорія та лексичне значення.

2.3.2 Тип та порядок

Бірд класифікує морфеми на флективні морфеми та похідні морфеми та стверджує, що ці два типи морфем поводяться по-різному при словотворенні[14].

Флективні морфеми утворюють закритий клас замість відкритого; він допускає нульові форми; воно взагалі не допускає подальшої трансформації словоформи; і, нарешті, його не можна кластеризувати парадигматично. Проте похідні морфеми не мають цих ознак.

Гіпотеза, згадана вище, переконливо свідчить про те, що морфологічний аналізатор повинен трактувати різні морфеми по-різному. Принаймні слід враховувати порядок застосування цих морфем. В утворенні словоформи в різний час беруть участь різні морфеми. Загалом послідовність може бути зафіксована в такому порядку: лексична морфема → похідна морфема → флективна морфема.

В англійській мові застосовується той самий порядок. Флективні афікси існують або в нульовій формі, або в кінці словоформи. Проте є три винятки з правила. Ці винятки наведені нижче:

- edly : abstractedly, admittedly, affectedly;
- ingly: agonizingly, amusingly, lingeringly, movingly;
- edness: bullheadedness.

Під час аналізу, протилежного формуванню форми, ми використовуємо зворотну послідовність: флективні морфеми → похідні морфеми → лексичні морфеми.

Точна процедура, яка виконується в цій системі, показана на рисунку 2.1.

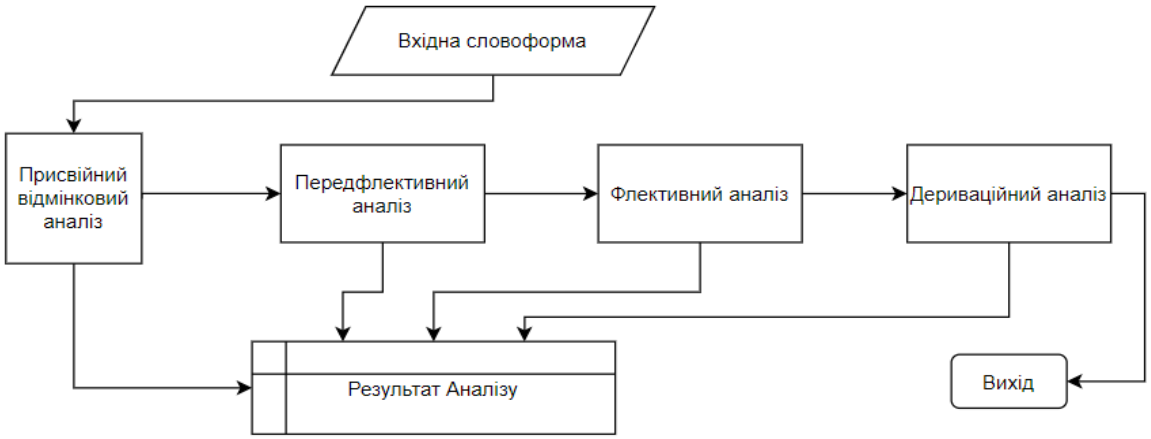

Рисунок 2.1 – Блок-схема для морфологічного аналізу, що є протилежністю словотвору

Було проведено експеримент, щоб оцінити продуктивність аналізатора, зображеного вище. Результати експерименту порівнюються з результатами в Unsupervised Segmentation of Words into Morphemes – Challenge 2005 (див. таблиця 2.1).

Таблиця 2.1 – Порівняння продуктивності різних морфологічних систем

Назва	Автор	Точність	Відкликання	F-оцінка
RePortS	Pitler and Keshava, Univ. Yale, USA	76.2	77.4	76.8
Cheat-all	Atwell et al, Leeds and Helsinki	86.0	70.4	77.4
Cheat-top5	Atwell et al, Leeds and Helsink	83.2	74.6	78.6
		88.46	78.61	83.24

Як показано в таблиці, аналізатор отримав досить кращий результат, ніж інші алгоритми. Помітно, що алгоритм перевершив алгоритм Кешава та Пілтер, на якому базується даний аналізатор.

Можна знайти дві причини такого покращення продуктивності. Перший – це усунення неоднозначності в морфологічному аналізі. Вивчення правил афіксів і процедура аналізу важливі для загальної ефективності морфологічного аналізу. Це переконання призвело до ретельного розгляду неоднозначності в морфологічному аналізі та систематичного вивчення явища. Класифікація неоднозначності на перехресну неоднозначність і комбінаторну неоднозначність, пов’язана з класифікацією в китайській сегментації, дозволяє нам трактувати неоднозначності по-різному. Це має вирішальне значення для покращення загальної продуктивності аналізатора.

Другою причиною є специфіка мови, якій приділялась увага під час створення аналізатора. Оскільки на меті малось створення морфологічного аналізатора, спеціально розробленого для англійської мови, стратегією автоматично стає увага певній мові. Таким чином, враховується багато

специфічних для мови морфологічних особливостей, таких як порядок застосування морфем, винятки з правил та інші. Іншим проявом особливої мовної турботи є налаштування параметрів і словникова шкала. Як обговорюється в аналізі статистичної мови, продуктивність системи, заснованої на статистичному навчанні, значною мірою залежить від навчальних даних.

3 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ ТА ТЕХНОЛОГІЙ

3.1 Опис прийнятих проектних рішень

Для виконання практичної частини дослідження було обрано мову програмування Python. Так вона є однією з найпопулярніших мов програмування для машинного навчання, і це з ряду причин:

- простота вивчення: Python має простий синтаксис, який легко зрозуміти і вивчити. Це зробило мову дуже популярною для початківців, а також для досвідчених програмістів, які швидко можуть вивчити нові бібліотеки та фреймворки;
- багата екосистема: Python має велику кількість бібліотек та фреймворків, що дозволяє легко виконувати завдання, пов'язані з машинним навчанням. Найпопулярніші з них – NumPy, Pandas, TensorFlow, Keras, PyTorch, Scikit-learn та інші;
- гнучкість: Python – інтерпретована мова програмування, що дозволяє швидко виконувати код і відлагоджувати програми. Також Python може бути використаний як для маленьких, так і для великих проектів, що робить його гнучким;
- зручна робота з даними: Python має вбудовану підтримку для роботи з даними, що дозволяє легко і швидко обробляти та аналізувати дані;
- підтримка спільнотою: Python має велику та активну спільноту розробників, що постійно розширює екосистему бібліотек і фреймворків. Також, завдяки цій спільноті, користувачі можуть швидко знайти відповіді на свої запитання і проблеми.

Усі ці фактори зробили Python однією з найбільш популярних мов програмування для машинного навчання, та забезпечили йому широку підтримку серед програмістів у цій області.

Також одним з найважливіших факторів при виборі мови програмування було те, що одна з найпопулярніших бібліотек у українському NPL є pymorphy2 та може бути використана тільки при написанні на мові Python.

У якості середовища для виконання коду, було обрано Azure Databricks[15]. Azure Databricks – це хмарна платформа для аналізу даних та машинного навчання, яка поєднує в собі потужність Apache Spark та зручність Databricks. Ця платформа дозволяє легко і швидко виконувати завдання аналізу даних та машинного навчання, забезпечуючи високий рівень безпеки та надійності.

Azure Databricks має ряд переваг:

- масштабованість: Платформа може легко масштабуватись, що дозволяє використовувати її для обробки великих обсягів даних та складних завдань машинного навчання;
- швидкість: Apache Spark, який є основою Azure Databricks, забезпечує високу швидкість обробки даних, що дозволяє швидко виконувати складні операції над даними та отримувати результати;
- спрощений процес розробки: Azure Databricks надає можливість використовувати мови програмування, такі як Python, R та Scala, для розробки та виконання завдань машинного навчання;
- багата екосистема: Azure Databricks має велику кількість інтегрованих бібліотек та фреймворків для машинного навчання, таких як TensorFlow, PyTorch, Scikit-learn та інші;
- високий рівень безпеки: Azure Databricks забезпечує високий рівень безпеки за допомогою інтеграції з Azure Active Directory та можливістю налаштування рівнів доступу до даних.

Усі ці фактори зробили Azure Databricks популярною платформою для аналізу даних та машинного навчання в хмарі, що дозволяє користувачам легко та ефективно вирішувати свої завдання.

У якості бібліотеки для морфологічного аналізу слів з використанням машинного навчання, було обрано бібліотеку pymorphy2, що є популярною при аналізі слів української мови.

Pymorphy2 – це бібліотека морфологічного аналізу текстів для мови Python, яка дозволяє проводити операції зі словами та їх формами. Основними можливостями бібліотеки Pymorphy2 є:

− морфологічний аналіз: бібліотека дозволяє проводити морфологічний аналіз української та російської мов. За допомогою Pymorphy2 можна отримати інформацію про форму слова (рід, число, відмінок, час, спосіб, наказовий спосіб), відмінювання дієслів та іменників;

− лематизація: бібліотека дозволяє проводити лематизацію слів. Лематизація – це процес зведення слів до їх базової форми (леми). Наприклад, слово "пішов" буде зведене до леми "йти";

− визначення частин мови: за допомогою Pymorphy2 можна визначити частину мови слова;

− конвертація форм слова: бібліотека дозволяє конвертувати слова з однієї форми в іншу форму (наприклад, з множинного числа в однина або з давального відмінка в називний);

− розпізнавання іменованих сутностей: бібліотека дозволяє визначити, чи є слово іменованою сутністю, наприклад, іменем людини, місцем, назвою організації тощо;

− спеллчекінг: Pymorphy2 може використовуватися для виправлення помилок у словах шляхом визначення правильної форми слова.

Загалом, бібліотека Pymorphy2 є потужним інструментом для обробки текстів на українській та російській мовах.

Вона дозволяє проводити широкий спектр операцій зі словами та їх формами, що є корисним у багатьох сферах.

3.2 Внутрішня будова бібліотеки pymorphy2

3.2.1 Словники

У pymorphy2 використовуються словники з проекту OpenCorpora, спеціально оброблені для швидких вибірок. Вихідний словник з OpenCorpora є файл, в якому слова об'єднані в лексеми, як зображено на рисунку 3.1.

```
гривня      NOUN,anim,masc sing,nomn
гривні     NOUN,anim,masc sing,gent
гривні     NOUN,anim,masc sing,datv
гривню     NOUN,anim,masc sing,accs
гривнею    NOUN,anim,masc sing,ablt
гривні     NOUN,anim,masc sing,loct
гривні     NOUN,anim,masc plur,nomn
гривень    NOUN,anim,masc plur,gent
гривням    NOUN,anim,masc plur,datv
гривні    NOUN,anim,masc plur,accs
гривнями   NOUN,anim,masc plur,ablt
гривнях    NOUN,anim,masc plur,loct
```

Рисунок 3.1 – Файл лексем зі словники OpenCorpora

Лексема складається з усіх форм слова, причому кожної форми зазначена граматична інформація (тег). Першою формою у списку йде нормальна форма слова.

Дві основні операції, які вміє робити морфологічний аналізатор – розбір та відмінювання слів. Якщо у нас є словник з лексемами, і ми хочемо розібрати/просхиляти словникове слово, ці операції дуже прості:

– розібрати слово – знайти його у словнику та повернути приписану йому граматичну інформацію;

– прохиляти слово – знайти слово у словнику, визначити його лексему, а потім знайти у лексемі потрібне слово із запитаними граматичними характеристиками.

Якщо все, що потрібно – розбір словникових слів, то можна завантажити всі слова та їхню граматичну інформацію на згадку “як є”, або зберегти в якусь БД загального призначення. Із цим є 2 проблеми:

– у словнику OpenCorpora близько 400тыс. лексем та 5млн окремих слів; якщо все завантажити в живлення list, то витратимо приблизно 2Гб оперативної пам'яті (у dict – ще більше);

– хотілося б, щоб операції з аналізу та відмінювання слів здійснювалися швидко – зокрема для слів, які у словнику, і слів, записаних якимось “спрощеним” методом.

Розглянемо лексему слова "розумний". Кожне слово в лексемі можна розбити

на 3 частини – "префікс", "стем" та "хвіст" (дви. рис. 3.2).

ПРЕФІКС	СТЕМ	ХВІСТ	ТЕГ
	розум	ний	ADJF,Qual masc,sing,nomn
	розум	ного	ADJF,Qual masc,sing,gent
		...	
	розум	ні	ADJS,Qual plur
	розум	ніше	COMP,Qual
	розум	ніший	COMP,Qual V-ej
над	розум	ний	COMP,Qual Cmp2,V-ej

Рисунок 3.2 – Лексеми слова “розумний” із розбиттям на префікс, стем та хвіст

"Стем" тут – частина слова, загальна всім слів у лексемі. Відкинемо його (див. рис. 3.3)

ПРЕФІКС	ХВІСТ	ТЕГ
	ний	ADJF,Qual masc,sing,nomn
	ного	ADJF,Qual masc,sing,gent
	...	
	ні	ADJS,Qual plur
	ніше	COMP,Qual
	ніший	COMP,Qual V-ej
над	ний	COMP,Qual Cmp2,V-ej

Рисунок 3.3 – Лексеми слова “розумний” із розбиттям на префікс та хвіст

По цій новій таблиці можна схиляти як слово “розумний”,так й інші слова – наприклад, “державний”. Отримана таблиця – є зразком для відмінювання слів.

При компіляції словника OpenCorpora pymorphy2 кожної лексеми визначає її парадигму. Виходить приблизно 3 тисячі унікальних парадигм (з приблизно 400 тисяч лексем).

Маючи парадигми, не потрібно зберігати всі лексеми та інформацію про те, до якої лексеми належить слово – достатньо зберегти парадигми та інформацію про те, за якою парадигмою слово змінюється.

Щоб зберігати парадигми більш компактно, вони перетворюються на масиви

чисел. Префіксам, "хвостам" та тегам присвоюються номери, і в парадигмах зберігаються лише ці номери. Рядки з префіксами, хвостами і тегами зберігаються окремо, в живих list. Номер рядка – це просто її індекс. Приклад закодованої в такий спосіб парадигми можна побачити на рисунку нижче (див. рис. 3.4).

prefix_id	suffix_id	tag_id
0	66	78
0	67	79
	...	
0	37	94
0	82	95
0	121	96
1	82	97
1	121	98

Рисунок 3.4 – Приклад закодованої парадигми

Кожна парадигма упаковується в одновимірний масив (array.array): спочатку йдуть усі номери хвостів, потім усі номери тегів, потім усі номери префіксів: 66 67 ... 37 82 121 82 121 | 78 79 ... 94 95 96 97 98 | 0 0 ... 0 0 0 1 1

Нехай парадигма складається із N форм слів; у масиві буде тоді N*3 елементів. Дані про і-й формі можна отримати за допомогою індексної арифметики: наприклад, номер граматичної інформації для форми з індексом 2 (індексація з 0) лежатиме в елементі масиву з номером N + 2, а номер префіксу для цієї форми – в елементі N *2 + 2.

У словнику OpenCorpora доступна інформація про зв'язки між лексемами. Наприклад, може бути пов'язана лексема для інфінітиву та лексему з формами дієслова, що відповідають цьому інфінітиву. Або, наприклад, форми короткого та повного прикметника.

Ця інформація дозволяє схиляти слова між частинами мови (наприклад, дієприкметник до дієслова).

У pymorphy2 всі пов'язані лексеми просто поєднуються в одну велику лексему на етапі підготовки (компіляції) вихідного словника; у скомпілюваному словнику інформація про зв'язки між лексемами явно недоступна.

Для зберігання даних про слова використовується кінцевий автомат (Deterministic Acyclic Finite State Automaton) з використанням бібліотек DAWG[16] (це обгортка над бібліотекою C++ dawgdic) або DAWG-Python (це написана на python реалізація DAWG, яка не вимагає компілятора для встановлення і працює швидше DAWG під PyPy).

У структурі даних DAFSA деякі загальні частини слів не дублюються і як результат потрібно менше пам'яті. Крім того, в DAWG можна швидко виконувати не тільки точний пошук слова, але й інші операції – наприклад, пошук префіксу або пошук із замінами.

У pymorphy2 в DAWG містяться не самі слова, а рядки виду: <слово> <розділювач> <номер парадигми> <номер форми у парадигмі>

3.2.2 Розбір не словникових слів

У тих випадках, коли слово не вдається знайти простим пошуком за словником, у справу вступають "провісники" – правила розбору не словникових слів.

У багатьох мовах існує набір словотворчих префіксів, які можна приписати до слова, і які при цьому не змінюють те, як слово розбирається та схиляється.

У pymorphy2 для кожної підтримуваної мови зберігається невеликий список таких префіксів (наприклад, "не", "псевдо", "супер", і т.д.). Якщо слово починається з одного з таких префіксів, то pymorphy2 відсікає префікс, розбирає те, що залишилося, а потім приписує префікс назад.

Якщо два слова відрізняються тільки тим, що до одного з них щось приписано спереду, то, швидше за все, вони будуть схилятися однаково. Тому якщо слово можна як префікс + якесь інше слово зі словника, то pymorphy2 вважає, що слово розбирається так само, як і це інше слово.

При цьому має виконуватися кілька додаткових умов:

- довжина словникового слова має бути не менше 3;

- довжина префікса не повинна бути більшою за 5;
- словникове слово – це іменник, прикметник, дієслово, дієприкметник або дієприслівник.

Робота алгоритму є такою: спочатку pymorphy2 намагається вважати першу букву префіксом, потім перші дві букви, потім перші три букви і т.д. до 5, і дивиться, чи немає залишку у словнику.

У підходах з відсіканням префіксів є два важливі обмеження:

- розбір не повинен залежати від префіксу (що невірно для словозмінних префіксів "по" та "най", які утворюють форми прикметників);
- морфологічний аналізатор повинен уміти розбирати праву частину слова (шляхом пошуку за словником чи ще якось) – права частина слова повинна мати якийсь сенс сама собою.

Розбір багатьох слів не можна передбачити, відсікаючи префікс та розбираючи решту як словникове слово.

Для того, щоб прогнозувати форми слів по тому, як слова закінчуються, при конвертації словників pymorphy2 збирає статистику по закінченням: для кожного можливого закінчення слова (від 1 до 5 букв) зберігаються всі можливі розбори. Іншими словами, кожному можливому 1..5-літерному закінченню зіставляється масив з інформацією про можливі варіанти аналізу (частота, номер парадигми, номер форми в парадигмі).

Якщо для кожного "закінчення" зберігати всі можливі варіанти аналізу, то вийде свідомо багато зайвих (дуже малоймовірних) правил. Тому отримані розбори "очищаються":

- pymorphy2 зберігає лише найчастіший розбір кожної частини промови;
- розбори, що належать “непродуктивним” парадигмам, видаляються (непродуктивними нині вважаються парадигми, яким відповідає менше лексем у словнику);
- рідкісні закінчення видаляються (ті, що зустрілися лише min_ending_freq=1 раз);

− не всі частини мови продуктивні: наприклад, не можна приписати щось до прийменника, щоб отримати інший привід; всі прийменники є у словнику, і пророкувати незнайомі слова як прийменники неправильно – такі варіанти пророкування відкидаються провісником.

Результат кодується у DAFSA. Схема зберігання схожа на ту, що в основному словнику, тільки:

− замість самих слів зберігаються всі їхні можливі закінчення;

− до номера парадигми та індексу форми в парадигмі додається ще "продуктивність" цього правила – кількість слів у словнику, які мають дане закінчення та розбираються цим чином <кінець слова> <розділювач> <продуктивність> <номер парадигми> <номер форми у парадигмі>.

Розбір зводиться до пошуку найбільш довгої правої частини слова, що розбирається, яка є в DAFSA із закінченнями.

Крім того, для кожного словозмінного префікса (ПЗ, НАІ) так само будується ще по одному DAFSA; якщо слово починається з одного з цих префіксів, то аналізатор додає результату варіанти передбачення, отримані пошуком за відповідним DAFSA.

pymorphy2 підтримує складові слова із двох частин, розділених дефісом. Для таких слів pymorphy2 спочатку розбирає обидві частини окремо (вони можуть бути не словниковими словами).

На даний момент підтримується 2 способи освіти таких слів:

Ліва частина – незмінна приставка/основа (наприклад "інтернет-магазин"). І тут форма слова визначається другою частиною. Цей випадок додається у можливі варіанти розбору завжди.

Рівноправні частини, що схиляються разом (наприклад, “людина-павук”). Цей випадок додається у можливі варіанти розбору лише тоді, коли обидві частини мають сумісну форму (є варіант розбору першої частини, який не суперечить якомусь варіанту розбору другої).

Однолітерні токени у верхньому регістрі pymorphy2 передбачає як ініціали: для них повертаються варіанти розбору "ім'я" та "по батькові", по всіх родах,

відмінках і числах.

При передбаченні до кінця слова результати сортуються за “продуктивністю” варіантів аналізу: найбільш продуктивні варіанти будуть першими.

Інакше кажучи, варіанти розбору(номери парадигм) упорядковані за частотою, з якою ці номери парадигм відповідають даному закінченню цієї частини промови – не враховуючи частотності по корпусу.

Експериментального підтвердження правильності цього підходу немає, але логіка тут така:

- нам не важливо, які слова у корпусі зустрічаються часто, так як. провісник працює для рідкісних слів, і рідкісні слова він передбачає як рідкісні, а не поширені;
- для “довгого хвоста” частотності у корпусі конкретні цифри мають дуже багато значення, так як флуктуації дуже великі;
- з іншого боку, важливо, які парадигми у українській мові більш продуктивні, які породжують більше слів.

Тому використовується частотність за парадигмами, отримана виключно зі словника.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Створення оточення для програмної реалізації

Першим етапом при створенні програмної реалізації є створення оточення. У нашому випадку це буде створення ресурсної групи, Azure Databricks сервісу та налаштування внутрішніх компонентів, таких як кластер та встановлення необхідної для дослідження бібліотеки pymorphy2.

Для створення ресурсної групи було вказано такі параметри (див. рис. 4.1):

- subscription: Visual Studio Professional Subscription;
- resource group: pymorphy2-analyse-rg01;
- region: (Europe) Poland Central.

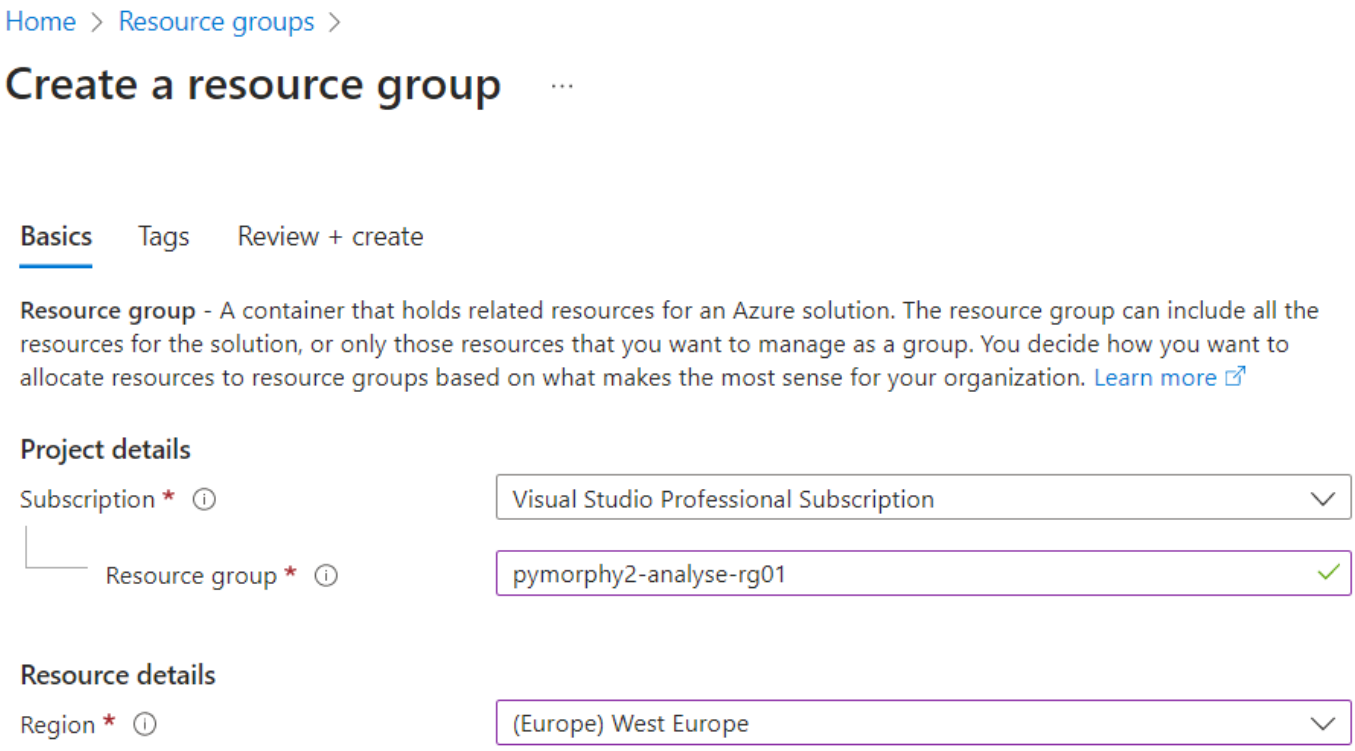

Рисунок 4.1 – Створення ресурсної групи

Наступним кроком є створення Azure Databricks сервісу та його налаштування. Для цього необхідно зайти в Azure Marketplace та знайти там Azure Databricks сервіс (див. рис. 4.2).

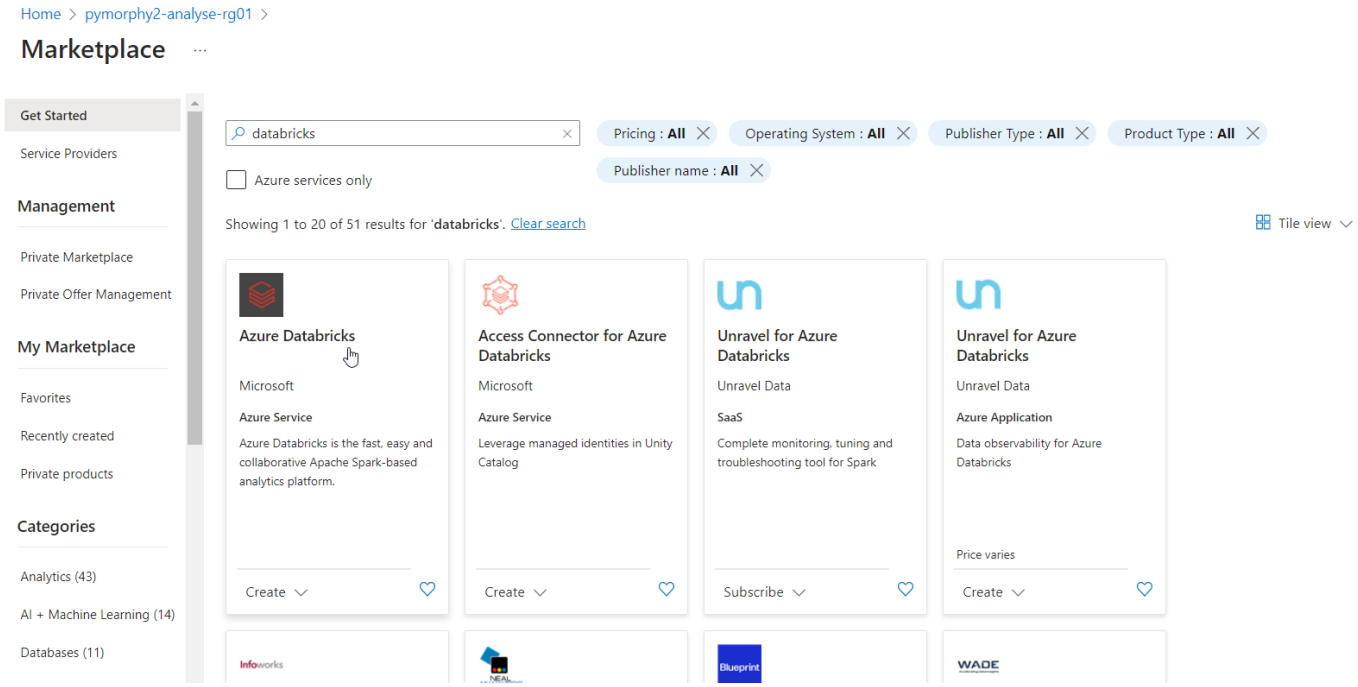

Рисунок 4.2 – Azure Databricks сервіс в Azure

Далі було створено Azure Databricks сервіс з описаними нижче параметрами:

- subscription: Visual Studio Professional Subscription;
- resource group: pymorphy2-analyse-rg01;
- region: North Europe;
- pricing Tier: Trial (Premium – 14-days Free DBUs).

Залишаємо дефолтні налаштування для мереж та шифрування. Вони не мають ніякого значення для нашого дослідження (див. рис. 4.3):

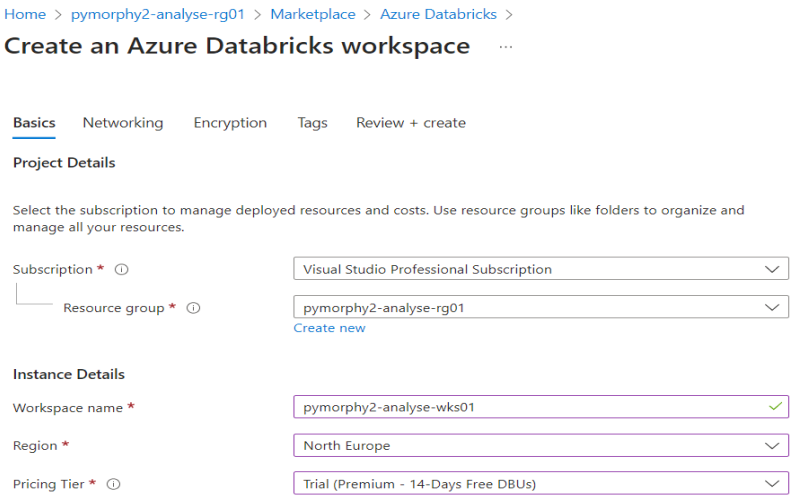

Рисунок 4.3 – Azure Databricks сервісу

У самому сервісі Azure Databricks є можливість перейти до робочої області, де було створено кластер на вкладці обчислень, для того, щоб створити кластер віртуальних машин, на яких виконуються всі обчислення та виконується код застосунків написаних у ноутбуках. Кластер було створено з наступними параметрами (див. рис. 4.4):

– policy: Unrestricted;

– access mode: Single user;

– single user access: Oleksandr Kovalov;

– databricks runtime version: 12.2 LTS (обираємо не останню версію, а саме LTS, щоб уникнути непередбачуваних проблем);

– worker type: Standard_DS3_v2 14 GB memory, 4 Cores;

– driver type: Same as worker ;

– min workers: 1;

– max workers: 2;

– enable autoscaling: yes (кількість воркерів буде збільшена з 1го до 2х у разі нестачі пам’яті);

– terminate after: 120 minutes (задля того, щоб уникнути зайвих трат коли кластер не використовується).

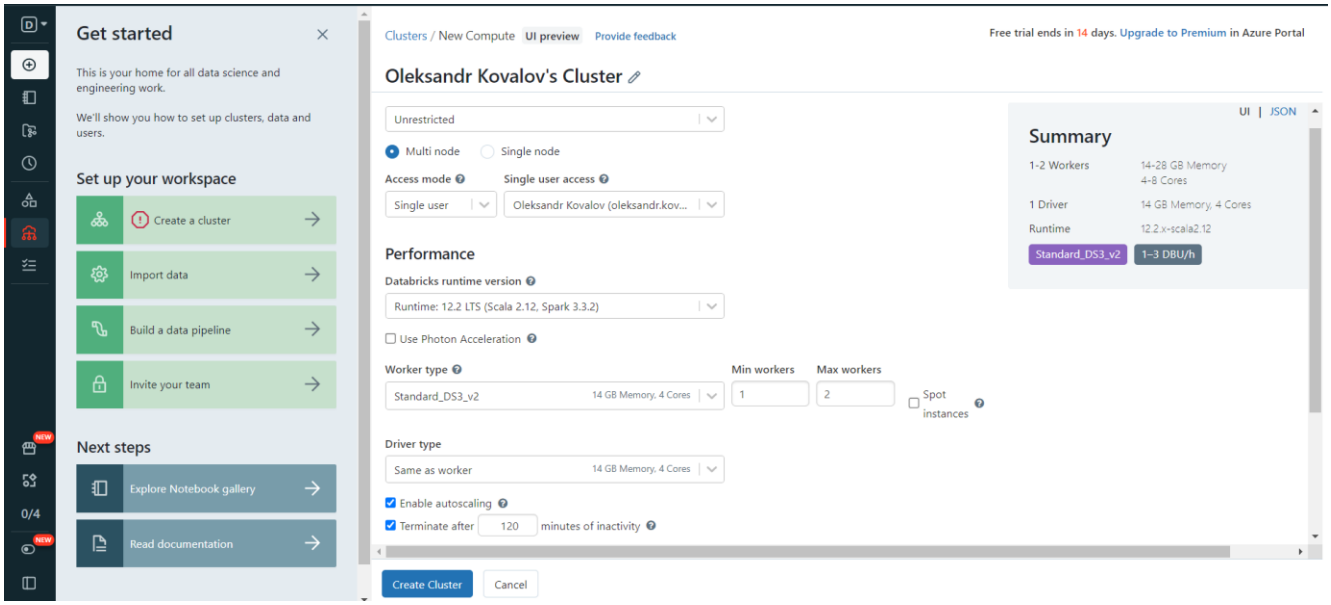

Рисунок 4.4 – Створення кластеру в Azure Databricks

Далі після успішного розгортання та запуску кластеру було встановлено дві PyPi бібліотеки (див. рис. 4.5):

– pymorphy2, як основна бібліотека, що використовувалася у досліджені, та була описана у попередніх розділах;

– pymorphy2-dicts-uk, додаткова бібліотека зі словником українських лексем, що дає можливість аналізу слів української мови.

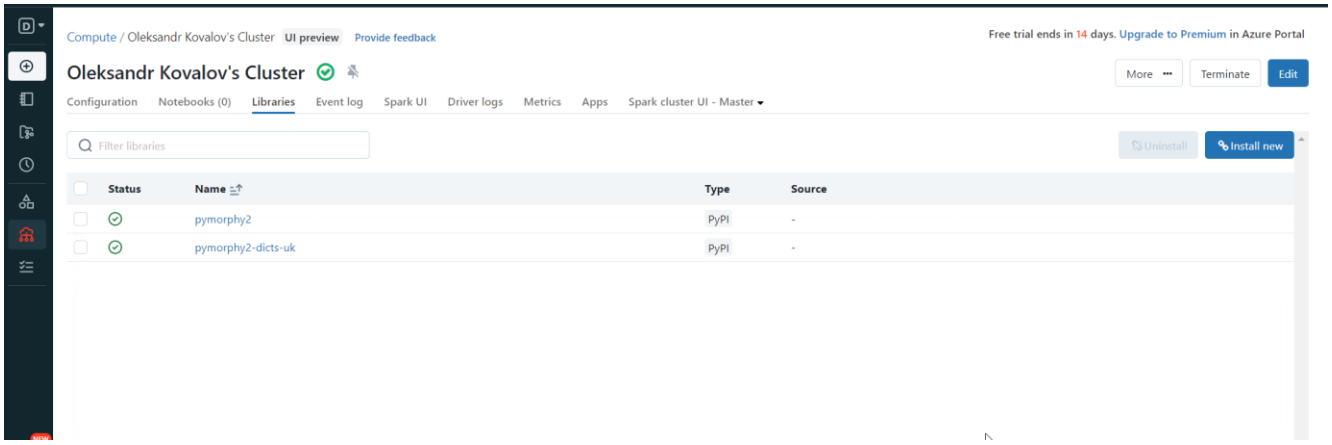

Рисунок 4.5 – Бібліотеки встановлені у Azure Databricks кластер

Бібліотеки також можуть бути використані локально задля цього у консолі необхідно виконати команди: pip install pymorphy2 та pip install -U pymorphy2-dicts-uk (див. рис. 4.6).

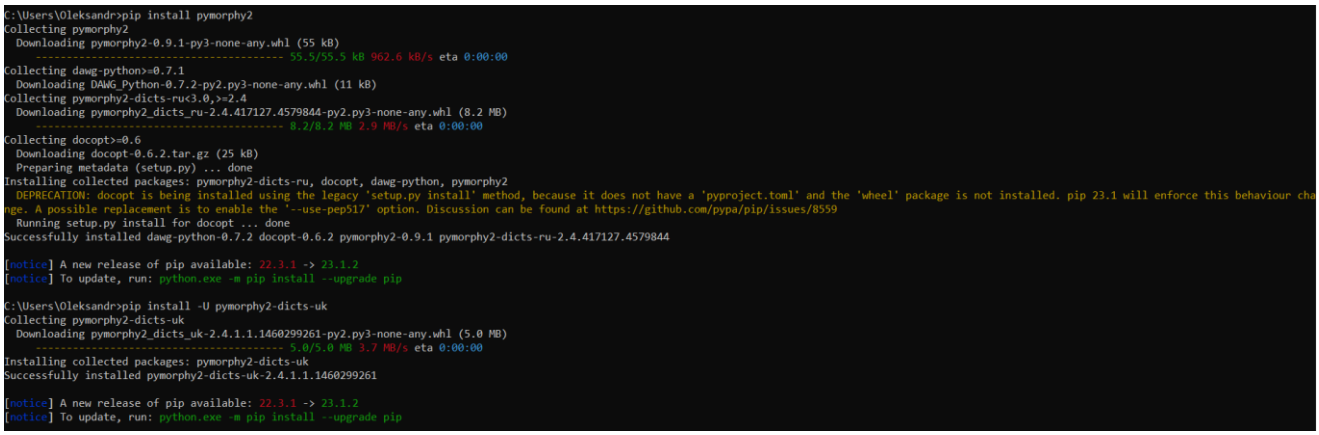

Рисунок 4.6 – Встановлення pymorphy2 бібліотек для локальної роботи

Останнім кроком у підготовці оточення є створення python ноутбуку, у якому написано код для дослідження роботи з pymorphy2 бібліотекою (див. рис. 4.7).

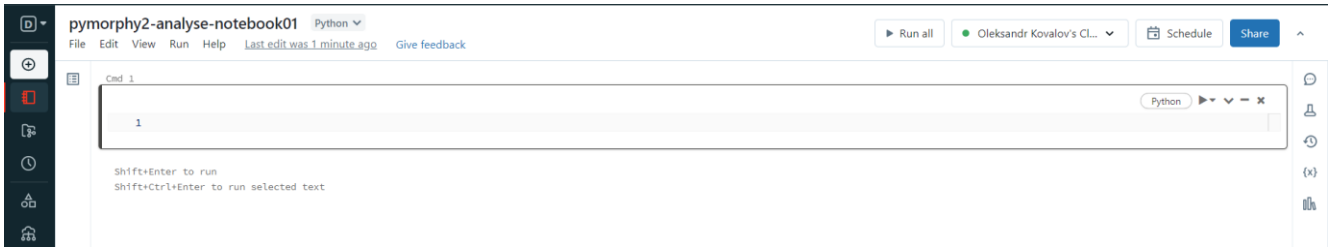

Рисунок 4.7 – Створення ноутбуку в Azure Databricks сервісі

Ноутбук має бути підключеним до створеного та запущеного кластеру, задля того, щоб мати можливість інтерпретувати та виконати написаний у ноутбуці код.

4.2 Реалізація програмного застосунку

Для роботи з бібліотекою та використання її методів у застосунку, першим кроком є імпорт бібліотеки pymorphy2 та створення об’єкту аналізатора (див. рис. 4.8).

```
1   # Підключення бібліотеки та створення об'єкта аналізатора
2   import pymorphy2
3   morph = pymorphy2.MorphAnalyzer(lang='uk')
```

Рисунок 4.8 – Підключення бібліотеки та створення аналізаторами

За допомогою метода MorphAnalyzer.parse() можна зробити аналіз окремого слова (див. рис. 4.9).

```
1   # Морфологічний аналіз слова на основі словника
2   morph.parse('мила')

Out[32]: [Parse(word='мила', tag=OpencorporaTag('NOUN,inan neut,gent'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 22, 1),)),
 Parse(word='мила', tag=OpencorporaTag('NOUN,inan plur,nomn'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 22, 8),)),
 Parse(word='мила', tag=OpencorporaTag('NOUN,inan plur,accs'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 22, 11),)),
 Parse(word='мила', tag=OpencorporaTag('NOUN,inan plur,voct'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 22, 14),)),
 Parse(word='мила', tag=OpencorporaTag('NOUN,femn,inan nomn'), normal_form='мила', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 61, 0),)),
 Parse(word='мила', tag=OpencorporaTag('ADJF,compb femn,nomn'), normal_form='милий', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 76, 9),)),
 Parse(word='мила', tag=OpencorporaTag('ADJF,compb femn,voct'), normal_form='милий', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 76, 15),)),
 Parse(word='мила', tag=OpencorporaTag('VERB,impf femn,past'), normal_form='мити', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 121, 20),)),
 Parse(word='мила', tag=OpencorporaTag('NOUN,anim femn,nomn'), normal_form='милий', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 624, 8),)),
 Parse(word='мила', tag=OpencorporaTag('NOUN,anim femn,voct'), normal_form='милий', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 624, 14),))]
```

Рисунок 4.8 – Морфологічний аналіз слова

На виході метод повертає один або декілька об’єктів типу Parse, з інформацією про те як слово може бути розібране. У нашому випадку було проаналізовано слово “мила”. Це слово може бути розібране, як іменник у декількох відмінках, дієслово та прикметник, що й можна побачити, дивлячись на об’єкти, що повернув аналізатор.

У кожного розбору слова є тег, що характеризує конкретний розбір слова. Наприклад, тег 'VERB,perf,intr plur,past,indc' означає, що слово – дієслово (VERB) доконаного виду (perf), неперехідний (intr), множини (plur), минулого часу (past), дійсного способу (indc).

Також бібліотека має функціонал для отримання характеристики кожного тегу. Наприклад отримання інформації тільки про частину мови, відмінок число або спосіб дієслова.

Для аналізу роботи цього функціоналу повернемо характеристику тегів для розбору слова “мила”, коли слово виступає іменником у родовому відмінку (див. рис. 4.9).

```
# Опис тегів (іменник)
parsed = morph.parse('мила')
print(f"Усі теги окремого розбору: {parsed[0].tag}")

print(f"Частина мови: {parsed[0].tag.POS}") #іменник
print(f"вид(доконаний чи недоконаний): {parsed[0].tag.aspect}")
print(f"Відмінок: {parsed[0].tag.case}") #родовий
print(f"Рід: {parsed[0].tag.gender}") #середній
print(f"способи дієслова(дійсний, умовний і наказовий): {parsed[0].tag.mood}")
print(f"Число: {parsed[0].tag.number}")
print(f"Лице(1,2,3): {parsed[0].tag.person}")
print(f"Час: {parsed[0].tag.tense}")
```

```
Усі теги окремого розбору: NOUN,inan neut,gent
Частина мови: NOUN
вид(доконаний чи не доконаний): None
Відмінок: gent
Рід: neut
способи дієслова(дійсний, умовний і наказовий): None
Число: None
Лице(1,2,3): None
Час: None
```

Рисунок 4.9 – Опис тегів слова “мила”, коли слово виступає іменником

Спочатку виконаний код вивів увесь тег, а потім кожну характеристику тегу окремо на новому рядку. Із отриманою інформацію можна побачити, що слово має такі характеристики:

– частина мови: іменник;

– відмінок: родовий;

– рід: середній.

Усі інші характеристики, такі як вид, спосіб дієслова, число, лице, час мають значення None, що говорить про те, що дані характеристики відсутні для розбору цього слова.

Також розглянемо тег для слова мити вже, коли слово виступає дієсловом (див. рис. 4.10).

```
# Опис тегів (дієслово)
parsed = morph.parse('мила')
print(f"Усі теги окремого розбору: {parsed[7].tag}")

print(f"Частина мови: {parsed[7].tag.POS}") #дієслово
print(f"вид(доконаний чи недоконаний): {parsed[7].tag.aspect}") #недоконаний
print(f"Відмінок: {parsed[7].tag.case}")
print(f"Рід: {parsed[7].tag.gender}") #жіночий
print(f"способи дієслова(дійсний, умовний і наказовий): {parsed[7].tag.mood}")
print(f"Число: {parsed[7].tag.number}")
print(f"Лице(1,2,3): {parsed[7].tag.person}")
print(f"Час: {parsed[7].tag.tense}") #минулий
```

```
Усі теги окремого розбору: VERB,impf femn,past
Частина мови: VERB
вид(доконаний чи не доконаний): impf
Відмінок: None
Рід: femn
способи дієслова(дійсний, умовний і наказовий): None
Число: None
Лице(1,2,3): None
Час: past
```

Рисунок 4.10 – Опис тегів слова “мила”, коли слово виступає дієсловом

Із отриманою інформацію можна побачити, що слово має такі характеристики:

– частина мови: дієслово;

– вид: недоконаний;

– рід: жіночий;

– час: минулий.

Усі інші характеристики, такі як відмінок, спосіб дієслова, число та лице мають значення None, що говорить про те, що дані характеристики відсутні для розбору цього слова.

Бібліотека дає можливість привести слово до нормальної форми (див. рис. 4.11).

```
#Нормальна форма
parsed = morph.parse('мила')

print(f"нормальна форма першого розбору: {parsed[0].normal_form}")

uniqueForms= []
for p in parsed:
    if p.normal_form not in uniqueForms:
        uniqueForms.append(p.normal_form)

print(f"усі унікальні нормальні форми слова:")
for form in uniqueForms:
    print(form)
```

```
нормальна форма першого розбору: мило
усі унікальні нормальні форми слова:
мило
мила
милий
мити
```

Рисунок 4.11 – Приведення до нормальної форми

Тут ми спочатку отримали нормальну форму для першого розбору, а потім виділили усі унікальні форми для всіх розборів і отримали інформацію, що усіма можливими нормальними формами для слова “мила” є мило, мила, милий та мити.

Бібліотека також підтримує функціонал відмінювання слова та різноманітні зміну його словоформи. Так для прикладу іменник мило у родовому відмінку було змінено (див. рис. 4.12):

– у орудний відмінок;

– у орудний відмінок та множину;

– у чоловічий рід.

```
#Відмінювання слів
parsed = morph.parse('мила')[0]

print(parsed.inflect({'ablt'})) #орудний
print(parsed.inflect({'plur', 'ablt'})) #орудний, множина
print(parsed.inflect({'masc'})) #чоловічий рід (помилка)
```

```
Parse(word='милом', tag=OpencorporaTag('NOUN,inan neut,ablt'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'милом', 22, 4),))
Parse(word='милами', tag=OpencorporaTag('NOUN,inan plur,ablt'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'милами', 22, 12),))
None
```

Рисунок 4.12 – Відмінювання слова та зміна словоформи

Як результат, можна побачити що слово було змінено на:

– милом;

– милами;

– None, так як слово не може бути приведено до чоловічого роду.

Також pymorphy2 надає функціонал для отримання усіх лексем слова, як зображено на рисунку нижче (див. рис. 4.13).

```
# Лексеми слова
parsed = morph.parse('мила')[0]

parsed.lexeme
```

```
Out[57]: [Parse(word='мило', tag=OpencorporaTag('NOUN,inan neut,nomn'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мило', 22, 0),)),
 Parse(word='мила', tag=OpencorporaTag('NOUN,inan neut,gent'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 22, 1),)),
 Parse(word='милу', tag=OpencorporaTag('NOUN,inan neut,datv'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'милу', 22, 2),)),
 Parse(word='мило', tag=OpencorporaTag('NOUN,inan neut,accs'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мило', 22, 3),)),
 Parse(word='милом', tag=OpencorporaTag('NOUN,inan neut,ablt'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'милом', 22, 4),)),
 Parse(word='мілі', tag=OpencorporaTag('NOUN,inan neut,loct'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мілі', 22, 5),)),
 Parse(word='милу', tag=OpencorporaTag('NOUN,inan neut,loct'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'милу', 22, 6),)),
 Parse(word='мило', tag=OpencorporaTag('NOUN,inan neut,voct'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мило', 22, 7),)),
 Parse(word='мила', tag=OpencorporaTag('NOUN,inan plur,nomn'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 22, 8),)),
 Parse(word='мил', tag=OpencorporaTag('NOUN,inan plur,gent'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мил', 22, 9),)),
 Parse(word='милам', tag=OpencorporaTag('NOUN,inan plur,datv'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'милам', 22, 10),)),
 Parse(word='мила', tag=OpencorporaTag('NOUN,inan plur,accs'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 22, 11),)),
 Parse(word='милами', tag=OpencorporaTag('NOUN,inan plur,ablt'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'милами', 22, 12),)),
 Parse(word='милах', tag=OpencorporaTag('NOUN,inan plur,loct'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'милах', 22, 13),)),
 Parse(word='мила', tag=OpencorporaTag('NOUN,inan plur,voct'), normal_form='мило', score=1.0, methods_stack=((DictionaryAnalyzer(), 'мила', 22, 14),))]
```

Рисунок 4.13 – Отримання лексем слова

Якщо слово є не широковживаним та не присутнє у словнику OpenCorpora, для аналізу цього слова буде застосовано провісник, який працює на основі машинного навчання, та робота, якого, була описана у попередніх розділах.

Для прикладу морфологічного аналізу слова за допомогою провісника, було

виконано аналіз слова “громовиною” (див. рис. 4.14).

```
# Морфологічний аналіз слова провісником
morph.parse('громовиною')
```

```
Out[72]: [Parse(word='громовиною', tag=OpencorporaTag('NOUN,femn,inan ablt'), normal_form='громовина', score=0.5, methods_stack=((DictionaryAnalyzer(), 'виною', 61, 4),
(UnknownPrefixAnalyzer(score_multiplier=0.5), 'громо'))),
 Parse(word='громовиною', tag=OpencorporaTag('NOUN,inan femn,ablt'), normal_form='громовина', score=0.3168316831683684, methods_stack=((FakeDictionary(), 'громовиною', 2
6, 4), (KnownSuffixAnalyzer(min_word_length=4, score_multiplier=0.5), 'виною'))),
 Parse(word='громовиною', tag=OpencorporaTag('ADJF femn,ablt'), normal_form='громовин', score=0.1782178217821782, methods_stack=((FakeDictionary(), 'громовиною', 93, 13),
(KnownSuffixAnalyzer(min_word_length=4, score_multiplier=0.5), 'виною')))]
```

Рисунок 4.14 – Морфологічний аналіз слова провісником

Можна спостерігати, що аналізатор повернув 3 можливі варіанти розбору слова “громовиною”. У об’єкті Parse присутня властивість score, що вказує на ймовірність того, що саме цей розбір слова є правильним. Також об’єкти повертаються відсортовано, починаючи з того, який є найбільш ймовірним.

Останнім пунктом програмної реалізації є створення датасету задля перевірки продуктивності роботи бібліотеки.

Оскільки в інтернеті не велика кількість датасетів зі словами української мови[17], датасет було створено на основі розділу українського роману, шляхом використання регулярного виразу, задля вирізання окремих слів. Для кожного слова був викликаний метод parse бібліотека pymorphy2, виконані заміри часу та зафіксована кількість сформованих об’єктів розбору слова(див. рис. 4.15).

```
dataset = re.findall(r'(?<!\S)[А-Яа-яІі]+(?!\S)|(?<!\S)[А-Яа-яІі]+(?=:(?!\S))', text)

parsedModels= []

start_time = time.time()

for word in dataset:
    parsedModels.append(morph.parse(word))

end_time = time.time()
time_lapsed = end_time - start_time

pCount = 0
for p in parsedModels:
    pCount += len(p)

print(f'кількість проаналізованих слів: {len(dataset)}');
print(f'Кількість згенерованих варіантів розбору: {pCount}');
print(f'час виконання: {time_lapsed} секунд');
```

```
кількість проаналізованих слів: 10593
Кількість згенерованих варіантів розбору: 33048
час виконання: 1.2418849468231201 секунд
```

Рисунок 4.15 – Аналіз продуктивності роботи бібліотеки

Як результат, ми можемо бачити, що час виконання аналізу 10593 слів тривав 1.2 секунди, та в результаті було сформовано 33048 Parse об’єктів розбору слова.

5 АНАЛІЗ ПРОВЕДЕНОГО ДОСЛІДЖЕННЯ

У результаті проведеного дослідження можна зробити аналіз та підбити деякі підсумки.

В ході аналізу предметної галузі було виявлено, що бібліотека pymorphy2 є найкращим варіантом для проведення морфологічного аналізу слів української мови. Більшість інших бібліотек взагалі не мають можливості для аналізу слів української мови, а бібліотека pymorphy2 має таку можливість за умови підключення необхідного словника та й може аналізувати не словникові слова за допомогою механізму провісника, побудованого на основі машинного навчання.

Було досліджено існуючі алгоритми, що використовуються для морфологічного аналізу слів на основі машинного навчання. Розглянуто 2 важливі аспекти з точки зору аналізу тексту, а саме усунення неоднозначностей та порядок правил афіксів.

Було розглянуто внутрішню будову та алгоритми роботи бібліотеки pymorphy2, як для слів, що вже присутні у словнику OpenCorpora, так і для не словникових слів із використанням машинного навчання.

У практичній частині дослідження були продемонстровані можливості та функції бібліотеки pymorphy2, а саме:

- морфологічний аналіз слова за словником;
- функціонал використання тегів;
- приведення слів до нормальної форми;
- відмінювання слова та зміна словоформи;
- отримання лексем слова;
- морфологічний аналіз слова провісником.

Останнім етапом дослідження була перевірка продуктивності бібліотеки pymorphy2 на основі створеного датасету, що показала те, що бібліотека має можливість проводити морфологічний аналіз 10000 слів за 1.2 – 1.5 секунди застосованій на створеному кластері з певною конфігурацією.

Для підбиття підсумку, можна сказати, що бібліотека pymorphy2 є потужним інструментом, і виправдовує свою популярність в українському NLP, але також можна виділити деякі недоліки:

- ймовірність розбору не словникового слова складає 79%, що може бути не достатньо у критично важливих частинах застосунків, що використовують дану бібліотеку;
- у бібліотеці є можливість зробити аналіз тільки окремих слів (часто значення слова дуже залежить від контексту, великим покращенням для бібліотеки був би аналіз слова з можливістю надавання слова у контексті речення);
- деякі слова у словнику OpenCorpora мають неоднозначності, що може завадити точному аналізу слова.

ВИСНОВКИ

На сьогодні існує велика кількість методів та бібліотек для проведення морфологічного аналізу слів для різних мов світу. Деякі з яких використовують й машинне навчання для проведення такого аналізу. Деякі підтримують аналіз слів Української мови, а деякі ні. Кожен застосунок чи бібліотека мають свої переваги та недоліки

У ході виконання курсової роботи, був проведений аналіз предметної галузі з метою ознайомлення з темою морфологічного аналізу слів та автоматичного аналізу і синтезу тексту. Було розглянуто найпопулярніші бібліотеки та застосунки для морфологічного аналізу тексту, а саме:

- pymorphy2;
- MyStem;
- phpmorphy;
- TreeTagger;
- FreeLing

У результаті дослідження було виявлено, що бібліотека pymorphy2 є оптимальним варіантом для проведення морфологічного аналізу слів української мови. Після чого було сформовано завдання самого дослідження, у результаті якого було розглянуто алгоритми для автоматичного морфологічного аналізу слів з використанням машинного навчання, розглянуто внутрішню будову роботи бібліотеки pymorphy2, продемонстровано увесь можливий функціонал бібліотеки, сформовано датасет та перевірено продуктивність роботи бібліотеки.

У результаті було проведено аналіз виконаного дослідження, виділено переваги та недоліки використаної бібліотеки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Автоматизований морфологічний аналіз тексту / URL: http://linguist.univ.kiev.ua/courses_morph.htm#:~:text=%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%B8%D0%B9%20%D0%BC%D0%BE%D1%80%D1%84%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%87%D0%BD%D0%B8%D0%B9%20%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D0%B7%20%D1%82%D0%B5%D0%BA%D1%81%D1%82%D1%83%20(%D0%90%D0%9C%D0%90,%D1%87%D0%B0%D1%81%2C%20%D0%B2%D0%B8%D0%B4%2C%20%D1%82%D0%BE%D1%89%D0%BE (дата звернення: 11.04.2023).

2. Як робити морфологічний розбір слова / URL: https://genomukr.com/nauka-j-osvitu/9357-jak-robiti-morfologichnij-rozbir-slova.html (дата звернення: 11.04.2023).

3. Nazarenko, D., Afanasieva, I., Golian, N., Golian, V. Investigation of the deep learning approaches to classify emotions in texts CEUR Workshop Proceedings, 2021, 2870, стр. 206–224

4. Tereshchenko, G., Gruzdo, I. Overview and Analysis of Existing Decisions of Determining the Meaning of Text Documents Tereshchenko, G., Gruzdo, I.2018 International Scientific-Practical Conference on Problems of Infocommunications Science and Technology, PIC S and T 2018 - Proceedings, 2019, стр. 645–653, 8632014 DOI 10.1109/INFOCOMMST.2018.8632014

5. What is Morphology / URL: https://study.com/academy/lesson/what-is-morphology-in-linguistics-definition-examples.html (дата звернення: 11.04.2023).

6. Морфологічний аналізатор pymorphy2 / URL: https://pymorphy2.readthedocs.io/en/stable/index.html (дата звернення: 11.04.2023).

7. phpMorphy / URL: https://phpmorphy.sourceforge.net/dokuwiki/ (дата звернення: 12.04.2023).

8. TreeTagger – a part-of-speech tagger for many languages / URL: https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/ (дата звернення: 14.04.2023).

9. FreeLing Home Page / URL: https://nlp.lsi.upc.edu/freeling/node/1 (дата звернення: 17.04.2023).

10. Goldsmiths, University of London / URL: https://www.gold.ac.uk/ (дата звернення: 17.04.2023).

11. CELEX2 / URL: https://catalog.ldc.upenn.edu/LDC96L14 (дата звернення: 18.04.2023).

12. Sajib Dasgupta, Vincent Ng. Unsupervised Morphological Parsing of Bengali / Published By: Springer, 2006 – pp. 331-330.

13. KIMMO / URL: https://software.sil.org/pc-kimmo/ (дата звернення: 23.04.2023).

14. Robert Beard. Lexeme-Morpheme Base Morphology: A General Theory of Inflection and Word Formation / Published By: Cambridge University Press, 1996 – pp. 497-504.

15. Azure DataBricks service / URL: https://learn.microsoft.com/en-us/azure/databricks/introduction/ (дата звернення: 25.04.2023).

16. DAWG / URL: https://dawg.readthedocs.io/en/latest/ (дата звернення: 28.04.2023).

17. Panchenko, Dmytro;Maksymenko, Daniil;Turuta, Olena;Yerokhin, Andriy;Daniiel, Yana;Turuta, Oleksii. Evaluation and Analysis of the NLP Model Zoo for Ukrainian Text Classification. Communications in Computer and Information Science, 2022, 1698 CCIS, pp. 109–123