Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем факультету інформатики

СТВОРЕННЯ NLP-СИСТЕМИ ДЛЯ ВИЗНАЧЕННЯ КОНТЕКСТУАЛЬНИХ СИНОНІМІВ УКРАЇНСЬКОЮ МОВОЮ

Текстова частина до кваліфікаційної роботи
за спеціальністю 121 «Інженерія програмного забезпечення»

Керівник кваліфікаційної роботи
ст.викл. Смиш О.Р.

_____
(підпис)
«____» _____ 2026 р.

Виконав студент ІПЗ-4
Швець Д.В.
«____» _____ 2026 р.

Київ — 2026

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри мультимедійних систем,

Доцент., к. ф.-м. н. О.П. Жежерун

_____(підпис)

«____» _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студентові 4 року навчання БП «Інженерія програмного забезпечення» факультету інформатики Швецю Дмитрові Віталійовичу

ТЕМА: Створення NLP-системи для визначення контекстуальних синонімів українською мовою

Зміст ТЧ до курсової роботи:

Анотація

Вступ

Огляд інструментів і наявних рішень

Розроблення системи для добору контекстуальних синонімів

Апробація отриманих результатів

Висновки

Список використаних джерел

Додатки

Дата видачі «___» _____ 2025 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Тема:

Створення NLP-системи для визначення контекстуальних синонімів українською мовою

Календарний план виконання роботи:

№ з/п	Назва етапу кваліфікаційної роботи	Термін виконання	Примітка
1	Отримання завдання на кваліфікаційну роботу	01.09.2025	
2	Огляд літератури за темою роботи	02.09.2025 — 01.04.2026	
3	Формування ідеї й архітектури проєкту	02.09.2025 — 15.09.2025	
4	Формування початкової збірки контекстуальних векторних представлень	16.09.2025 — 01.11.2025	
5	Очищення, дедуплікація та злиття сховищ векторів	02.11.2025 — 30.12.2025	
6	Розроблення модуля вилучення гіперонімів зі словникових тлумачень	16.12.2025 — 19.01.2026	
7	Розроблення методу добору контекстуальних синонімів	16.09.2025 — 31.01.2026	
8	Реалізація модуля морфологічного узгодження синонімів	01.11.2025 — 31.01.2026	
9	Створення вебзастосунку	01.11.2025 — 15.02.2026	
10	Оцінювання якості роботи системи	10.02.2026 — 10.03.2026	
11	Розроблення вдосконалень системи за результатами оцінювання	16.02.2026 — 09.03.2026	
12	Підготовка візуалізацій отриманих результатів	01.04.2026 — 13.04.2026	
13	Остаточне оформлення текстової частини та презентації	14.04.2026 — 14.05.2026	
14	Захист кваліфікаційної роботи	25.05.2026 — 27.05.2026	

Графік узгоджено «___» _____

Студент Швець Дмитро Віталійович

Керівник Смиш Олег Русланович

АНОТАЦІЯ

У роботі описано розроблення NLP-системи для визначення контекстуальних синонімів у текстах, написаних українською мовою. Сформовано збірку контекстуальних векторних представлень лексем на основі корпусів ГРАК, CNC-UA та UD-treebanks із застосуванням моделі paraphrase-multilingual-mpnet-base-v2. Вектори обчислено методом усередненого пулінгу прихованих станів субтокенів цільової леми в контексті речення.

На основі тлумачень СУМ побудовано збірник гіперонімів: для кожного запису засобами синтаксичного аналізу залежностей виявлено гіпероніми та виділено їх у тексті тлумачення.

Реалізовано модуль морфологічного узгодження синонімів із граматичним контекстом речення.

Розроблено два вдосконалення системи: формування синтаксично орієнтованого контексту для обчислення вектора цільового слова та переранжування кандидатів через порівняння ембедингів речень. Після обох удосконалень точність системи становить 68,28 %, а повнота — 49,18 %, що відповідно у 2,33 і 2,06 раза перевищує початкові значення.

Створено вебзастосунок для наочної взаємодії з функціями системи. Отримані результати візуалізовано.

Ключові слова: обробка природної мови, контекстуальні синоніми, текстовий корпус, векторне представлення слова, ембединг, семантична подібність, морфологічне узгодження, гіпероніми, українська мова, трансформерні моделі, вебзастосунок.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

ВЕСУМ	Великий електронний словник української мови
ГРАК	Генеральний регіональний анотований корпус
СУМ	Словник української мови
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
CNC-UA	A Contemporary News Corpus of Ukrainian
CoNLL-U	Computational Natural Language Learning
FAISS	Facebook AI Similarity Search
JSON	JavaScript Object Notation
MLM	Masked Language Modelling
NER	Named Entity Recognition
NLP	Natural Language Processing
PMMB2	paraphrase-multilingual-mpnet-base-v2
POS	Part of Speech
RoBERTa	Robustly Optimized BERT Approach
UD	Universal Dependencies
UMAP	Uniform Manifold Approximation and Projection
WSD	Word Sense Disambiguation
XLM	Cross-Lingual Language Model

ВСТУП

Актуальність теми

Одним із основних завдань у галузі обробки природної мови є забезпечення розуміння комп'ютером семантичного змісту тексту. Здатність автоматично визначати, які слова є взаємозамінними в певному контексті, лежить в основі цілого ряду практичних застосувань: автоматичного реферування, машинного перекладу, перевірки тексту на тавтологію тощо. Проте розуміння синонімії не зводиться до тривіального перелічення еквівалентів зі словника, адже те, яке слово є доречним замінником, принципово залежить від контексту.

Розроблення інструментів NLP для мов із обмеженим ресурсним забезпеченням є актуальним науковим напрямом. Порівняно з англійською мовою, для якої наявна розгалужена система готових рішень: тезаурусів, семантичних мереж, навчених моделей, — українська мова залишається менш охопленою. Флективна морфологія ускладнює формування навчальних корпусів і практичне застосування результатів: знайти синонім-лему недостатньо — потрібно увідповіднити його до правильної граматичної форми згідно з контекстом речення.

Тому актуальним є розроблення системи, що визначає контекстуальні синоніми для українськомовних текстів та повертає результати в граматично узгодженій формі, придатній для безпосереднього використання.

Мета і завдання дослідження

Мета кваліфікаційної роботи полягає в тому, щоб автоматизувати процес визначення контекстуальних синонімів у текстах, написаних українською мовою.

Виокремлено такі *завдання* кваліфікаційної роботи:

- дослідити наявні інструменти та рішення для оброблення українськомовних текстів і визначення семантичної подібності слів;

- сформувати збірку контекстуальних векторних представлень слів з українськомовних корпусів;
- виокремити гіпероніми лексем з їхніх словникових тлумачень, долучивши до сформованої збірки;
- розробити методи добору контекстуальних синонімів до обраного слова в реченні;
- реалізувати модуль морфологічного узгодження синонімів із граматичним контекстом;
- оцінити роботу системи та візуалізувати отримані результати.

Об'єктом дослідження є автоматичне оброблення семантики в текстах, написаних українською мовою.

Предметом дослідження є застосування контекстуальних векторних представлень для визначення синонімічних коваріантів слова в конкретному семантичному оточенні.

Методи дослідження

У роботі використано евристичний і емпіричний методи наукового дослідження. Застосовано методи системного підходу (аналізування, декомпозицію, синтезування), процедурного й об'єктно-орієнтованого програмування. Серед спеціальних методів використано токенізацію, лематизацію, граматичне позначення, а також зниження розмірності векторів.

Практичне значення отриманих результатів

Розроблено NLP-систему, що автоматично визначає контекстуальні синоніми для довільного слова в українськомовному тексті та повертає результати в граматично узгодженій формі. Система підтримує чотири частини мови: іменники, прикметники, дієслова та прислівники — і для кожної реалізує окремий механізм морфологічного узгодження.

Запропоновано метод формування збірки контекстуальних векторних представлень лексем на основі текстових корпусів із використанням синтаксично орієнтованого контексту, що враховує семантично релевантне

оточення слова в реченні. Розроблено метод переранжування кандидатів-синонімів через порівняння на рівні речень для підвищення точності системи.

Для розширення результатів системи вилучено гіпероніми на основі тлумачень СУМ із застосуванням синтаксичного аналізування залежностей.

Практичне застосування розробленої системи охоплює широке коло користувачів. Редактори, коректори та журналісти використовуватимуть систему для уникнення тавтологій і збагачення лексики текстів. Письменники та перекладачі отримують інструмент для пошуку стилістично точного замінника слова з урахуванням його конкретного вживання в реченні. Студенти та науковці — для підвищення різноманітності й точності мовлення в академічних роботах. Система послугує готовим компонентом NLP-орієнтованих застосунків для автоматичного реферування, перефразування, стилістичної перевірки тексту тощо. Наявність гіперонімів розширює можливості застосування системи за межі лексичної заміни — до змістового узагальнення тексту.

Структура й обсяг кваліфікаційної роботи

Кваліфікаційна робота містить вступ, три розділи, висновки, список використаних джерел і один додаток. Загальний обсяг кваліфікаційної роботи становить 63 сторінки. У роботі наявні 17 рисунків і 4 таблиці. Список використаних джерел містить 37 найменувань.

РОЗДІЛ 1.
ОГЛЯД ІНСТРУМЕНТІВ І НАЯВНИХ РІШЕНЬ

Розділ присвячений аналізуванню термінологічної бази, інструментів і наявних рішень у галузі обробки природної мови, що стосуються завдання визначення контекстуальних синонімів в українськомовних текстах. У підрозділі 1.1 визначено поняття, що стосуються кваліфікаційної роботи. У підрозділі 1.2 описано інструменти — корпуси, словники та мовні аналізатори, застосовні для оброблення текстів, написаних українською мовою, а також моделі векторного представлення. Підрозділ 1.3 містить огляд наявних рішень, суміжних із поставленим завданням: модель BERT з механізмом маскованого мовного моделювання, інструмент LanguageTool із функцією добору синонімів, а також обрана модель paraphrase-multilingual-mpnet-base-v2. Окремо описано метод дотренування моделі на завданні розрізнення омонімів, запропонований у роботі Laba та ін., та обґрунтовано доцільність його застосування.

1.1. Опис метамови галузі

Природна мова відрізняється від формалізованої: вона допускає неоднозначність, контекстну залежність значень і варіативність вираження одного й того ж змісту. Ці властивості роблять автоматичне оброблення текстів, написаних природною мовою, нетривіальним завданням, що потребує спеціальних методів і підходів.

Обробка природної мови (Natural language processing, NLP) — це міждисциплінарна галузь на перетині лінгвістики, інформатики та штучного інтелекту, що вивчає методи автоматичного аналізу, інтерпретації та генерування текстів, написаних людськими мовами [1].

Одним із центральних явищ, що ускладнюють автоматичне оброблення тексту, є багатозначність слів, засвідчена у двох основних формах: полісемії та

омонімії. Полісемія — це здатність одного слова мати декілька семантично споріднених значень, що утворюють єдину мережу смислів на основі спільного семантичного ядра [2, с. 194]. Омонімією ж називають явище, за якого два або більше слів збігаються за написанням та/або звучанням, але мають різні, семантично не пов’язані між собою значення [2, с. 198]. На відміну від полісемії, омоніми є принципово різними лексичними одиницями. Наприклад, «коса» (заплетене волосся) і «коса» (сільськогосподарське знаряддя) є омонімами — словами, що збігаються за формою, проте мають різні та семантично не пов’язані значення.

Синоніми — це слова, що мають значення, які повністю або частково збігаються. Слово вступає в синонімічні відношення окремими значеннями, тому багатозначне слово одночасно може належати до кількох синонімічних рядів [2, с. 202].

Контекстуальні синоніми — це слова, які набувають однакової або схожої семантики в межах конкретного тексту, але поза ним можуть мати різний зміст. Зближення їхніх значень ґрунтується на ситуативній суміжності та функціональній схожості, що уможливлює об’єднання початково різних понять у єдиний синонімічний ряд [3, с. 24].

Варто згадати про розрізнення значень слів (Word Sense Disambiguation, WSD) — завдання автоматичного визначення того, яке саме значення слова вжито в конкретному контексті [4, с. 11]. Для цього використано векторні представлення слів (word embedding, ембединг) — числові вектори фіксованої розмірності, що кодують семантичні та синтаксичні властивості слів на основі вживання у великих текстових корпусах. Слова з близьким значенням у просторі векторних представлень розташовані ближче одне до одного [4, с. 13].

Створена збірка містить саме контекстуальні ембединги — векторні представлення слів, що враховують семантичне оточення в реченні. На відміну від статичних представлень, де кожне слово отримує єдиний вектор незалежно від контексту, сучасні моделі, засновані на архітектурі трансформера, формують

окремий вектор (або декілька векторів) для кожного входження слова [5, с. 4172]. Це уможливлює розрізнення значень омонімів і полісемічних слів.

Розроблена система порівнює слова та речення за семантичною подібністю — мірою близькості значень двох мовних одиниць. У контексті NLP семантичну подібність обчислюють як косинусну подібність — міру схожості двох векторів, обчисленої як косинус кута між ними та набуває значень від −1 до 1, де 1 означає повний збіг напрямків [1].

1.2. Огляд інструментів

Для оброблення текстів, написаних українською мовою, доступний ряд інструментів: текстові корпуси, словники та мовні аналізатори. Їхнє застосування є основою для розроблення систем, що потребують розуміння лексики та семантики українськомовних текстів.

1.2.1. Корпуси та словники

Текстовий корпус — це комп’ютерне зібрання текстів відповідної мови, що призначене для наукового та практичного вивчення мови [6, с. 36].

Для формування навчальних і оцінювальних наборів даних в роботі використано три корпуси, що охоплюють різні функціональні стилі сучасної української мови.

Першим використано Генеральний регіональний анотований корпус (ГРАК) — це збалансований корпус сучасної української мови, що містить тексти різних жанрів і стилів [7]. Наявність морфологічної та синтаксичної анотації робить ГРАК придатним джерелом для формування контекстуальних векторних представлень слів у різних лексико-граматичних умовах.

До формування збірки залучено також Сучасний новинний корпус для української мови (CNC-UA) [8] — корпус новинних текстів публіцистичного

стилю. Використання корпусу новин забезпечує різноманітність лексичних контекстів.

Останнім корпусом, використаним у роботі, є UD-treebanks, що містить набори текстів, розмічених відповідно до принципів Universal Dependencies (UD) — уніфікованої нотації для синтаксичного аналізу, яка однаково інтерпретована для різних мов [9]. Завдяки синтаксичній розмітці ці дані уможливлюють не лише семантичний, а й морфосинтаксичний аналіз лексем у контексті.

Для валідації сформованих наборів даних застосовано Великий електронний словник української мови (ВЕСУМ) [10] — комплексний лексикографічний ресурс, що охоплює загальновживану лексику й містить морфологічну інформацію [11, с. 137]. Для отримання гіперонімів з дефініцій доцільно використовувати СУМ (Словник української мови) — академічний тлумачний словник Національної академії наук України, що охоплює лексику від кінця XVIII ст. до сьогодення [12].

Щоб оцінити роботу створеної системи, використано словник синонімів від uSofTrod — він містить понад 68 тис. реєстрових одиниць, понад 18 тис. синонімічних рядів [13], що уможливлює його використання для оцінювання роботи системи.

1.2.1. Мовні аналізатори

Для попереднього оброблення тексту — токенізації, лематизації, розмічування частин мови, визначення морфосинтаксичних зв’язків — доступний ряд мовних аналізаторів, що підтримують українську мову.

Токенізація — це операція поділу текстового потоку на дискретні одиниці — токени, якими зокрема є слова [14, с. 216].

Лематизація є методом зведення словоформи до її канонічного вигляду — леми, що забезпечує коректне зіставлення різних граматичних форм одного слова [1].

Розмічування частин мови (POS-tagging) — це автоматичне присвоєння кожному токену тексту граматичної категорії: іменника, дієслова, прикметника тощо [1].

UDPipe — це конвеєрний мовний аналізатор, здатний виконувати повний цикл лінгвістичного аналізу тексту: від сегментації на речення і токени до лематизації, POS-tagging і побудови синтаксичного дерева залежностей [15]. Модель підтримує донавчання на довільних UD-сумісних даних, що уможливлює її адаптацію до нових мов. Для оброблення українськомовних текстів UDPipe застосовує модель, натреновану на «золотому стандарті» — вручну анотованому корпусі Інституту української мови [16].

Stanza — це багатомовний NLP-пакет для Python, розроблений у Стенфордському університеті, що реалізує нейромережевий конвеєр аналізу тексту в рамках UD-специфікації [17]. Зокрема, Stanza використовується для POS-tagging лем у формуванні наборів даних для завдання WSD в українськомовних текстах [4, с. 12].

spaCy — це Python-бібліотека для NLP. Окрім стандартних операцій токенізації та синтаксичного аналізу, spaCy містить вбудовані засоби для виявлення іменованих сутностей (Named Entity Recognition, NER). Хоча синтаксична модель spaCy не є UD-сумісною, сформовані нею дерева залежностей можна узгодити з UD-нотацією [18].

1.3. Огляд наявних рішень

1.3.1. BERT і маскована мовна модель

BERT (Bidirectional Encoder Representations from Transformers) — це попередньо навчена нейромережева модель, що ґрунтується на архітектурі трансформера й формує двонаправлені контекстуальні векторні представлення токенів [5]. Основним механізмом попереднього навчання BERT є маскована мовна модель (Masked Language Modelling, MLM): під час навчання частину

токенів у вхідному тексті замінюють спеціальним символом, а модель навчається передбачати, яке слово мало стояти на місці маски, спираючись на оточувальний контекст [5, с. 4171].

Механізм MLM можна формально розглядати як інструмент для добору семантично близьких відповідників: якщо замаскувати цільове слово, модель запропонує набір найімовірніших токенів-кандидатів на його позицію. Однак такий підхід має обмеження в контексті пошуку синонімів. По-перше, BERT оптимізований на відтворення найімовірнішого слова в позиції маски з огляду на загальний розподіл у навчальному корпусі, а не на пошук семантично рівнозначних кандидатів серед визначеного словника синонімів. По-друге, BERT не призначений для ефективного попарного порівняння семантичної подібності між реченнями: для цього потрібно двічі пропускати пари речень через модель, що є обчислювально надмірним. По-третє, MLM не гарантує збереження вихідного змісту речення після заміни слова: запропонований кандидат може бути граматично і статистично ймовірним, але семантично не еквівалентним вихідному слову в контексті. З огляду на ці обмеження, BERT у режимі MLM не є оптимальним рішенням для завдання добору контекстуальних синонімів.

1.3.2. LanguageTool

LanguageTool [19] — це відкрите програмне забезпечення для перевірки граматики, стилю та орфографії, що підтримує понад 30 мов [20]. Серед можливостей інструменту передбачена функція для підбору синонімів: у режимі перефразування користувач отримує список лексичних замінників для виокремленого слова [21], як показано на рисунку 1.1.

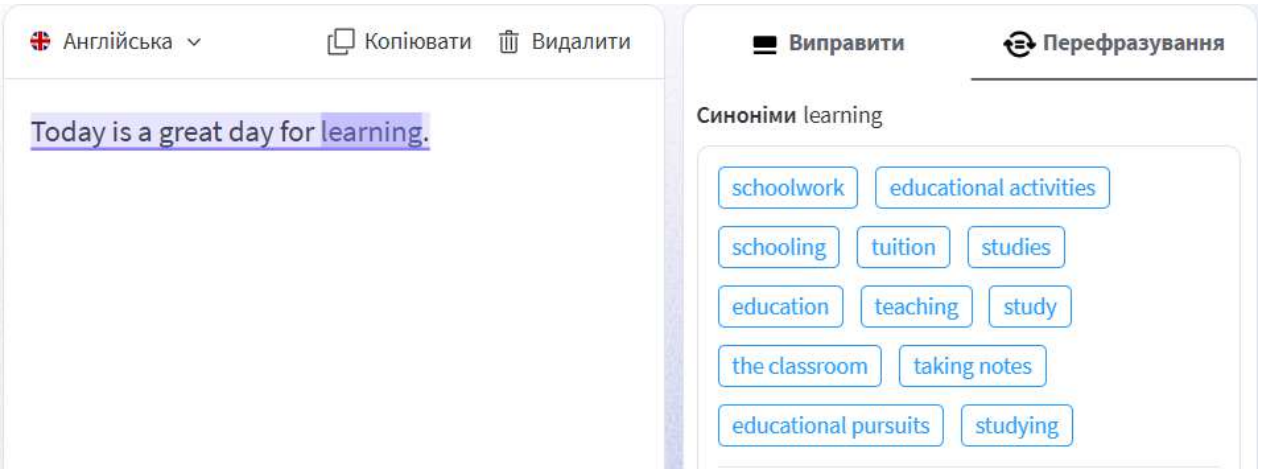

Рисунок 1.1. Підбір синонімів у LanguageTool англійською мовою

Примітно, що LanguageTool підтримує українську мову в частині граматичної та орфографічної перевірки [20]. Однак функція підбору синонімів є доступною лише для обмеженого набору мов [21]. Тобто саме той модуль, який концептуально найближчий до завдань цієї роботи, не підтримує української мови. Тож попри концептуальну кандидатуру LanguageTool як готового рішення для підбору синонімів із урахуванням граматичного контексту, його безпосереднє застосування до українськомовних текстів у частині перефразування є неможливим.

1.3.3. Модель paraphrase-multilingual-mpnet-base-v2

Для розроблення системи визначення контекстуальних синонімів обрано модель paraphrase-multilingual-mpnet-base-v2 (далі — PMMB2), розміщену у відкритому доступі на платформі Hugging Face [22].

PMMB2 належить до сімейства Sentence Transformers — моделей, навчених для отримання векторних представлень цілих речень, що відображають їхній семантичний зміст [23, с. 3982]. Архітектурно модель побудована на основі багатомовного трансформера XLM-RoBERTa [24] із забезпеченням оброблення текстів більш ніж 50 мовами, охопно з українською [25, с. 4515], — що є головною умовою для її застосування у кваліфікаційній роботі. Модель формує

векторні представлення розмірністю 768, застосовуючи стратегію усередненого пулінгу (mean pooling) для агрегації представлень токенів у репрезентацію цілого речення [25, с. 4513].

PMMB2 навчена на багатомовних наборах парафраз із застосуванням сіамської мережевої архітектури та функції втрат на основі семантичної подібності між реченнями [25, с. 4515]. Це означає, що семантично близькі речення різними мовами отримують близькі векторні представлення в єдиному семантичному просторі. Для визначення контекстуального синоніма можна обчислити векторне представлення речення з кожним словом-кандидатом і порівняти їх із представленням вихідного речення за допомогою косинусної подібності.

Вибір PMMB2 підтверджено емпірично: дослідження [4, с. 16] показало, що серед усіх протестованих багатомовних моделей (bert-base-multilingual-cased, xlm-roberta-base, xlm-roberta-large та ukr-roberta) саме PMMB2 зі стратегією mean pooling демонструє найвищу точність у завданні WSD для українськомовного набору даних — 73,5 % без дотренування.

1.3.4. Дотренування моделі для розрізнення омонімів

У згаданому вище дослідженні [4, с. 14] запропоновано метод дотренування (fine-tuning) моделі на спеціально сформованому наборі даних для завдання розрізнення омонімів в українській мові, що дає змогу підвищити точність семантичних представлень.

Набір даних для дотренування побудовано повністю неконтрольованим способом на основі корпусу UberText 2.0 [26]: для кожного омоніма з оцінювального набору відібрано речення, що містять відповідне слово, після чого за допомогою моделі обчислено косинусні відстані між векторними представленнями речень. Речення з найвищою схожістю використано як позитивний приклад (слово вжите в тому ж значенні, що й у вихідному реченні),

а речення з найнижчою схожістю — як негативний приклад (слово вжите в іншому значенні). Так сформовано набір із ~1,2 млн трійок «вихідне, позитивне, негативне речення».

Для дотренування застосовано функцію втрат TripletMarginLoss [27]: вона мінімізує відстань між векторним представленням вихідного речення та позитивного прикладу та водночас максимізує відстань між вихідним реченням і негативним прикладом. За результатами дослідження дотренована модель PMMB2 досягає 77,9 % загальної точності WSD на наборі омонімів, перевищуючи базову версію без дотренування [4, с. 16]. Також встановлено, що дотренована модель краще розрізняє менш уживані значення слів, тобто не зводиться до вибору найпопулярнішого значення. Ця властивість є важливою для завдання добору контекстуальних синонімів, де те саме слово можна вжити в рідшому, але семантично точному значенні. З огляду на це, модель PMMB2, дотреновану описаним методом, долучено до розробленої системи.

1.4. Висновки до розділу 1

У розділі систематизовано термінологічну базу, оглянуто інструменти та наявні рішення, суміжні з завданням визначення контекстуальних синонімів для українськомовних текстів.

З’ясовано, що для оброблення українськомовних текстів наявний ряд мовних аналізаторів: UDPipe, Stanza та spaCy, — а також відкриті корпуси (ГРАК, CNC-UA, UD-treebanks) і словники (СУМ, ВЕСУМ, словник синонімів від uSofTrod), що дають змогу сформувати збірку векторних представлень й оцінити роботу створеної системи.

Розглянуто механізм BERT на основі маскованого мовного моделювання та встановлено, що він не є оптимальним для поставленого завдання. Аналіз інструменту LanguageTool засвідчив, що, попри наявність функції добору синонімів, він не підтримує української мови.

Обґрунтовано вибір моделі paraphrase-multilingual-mpnet-base-v2 як основного інструменту: побудована на XLM-RoBERTa, підтримує українську мову, продемонструвала найвищу точність серед порівнюваних моделей у завданні WSD для українськомовних текстів — 73,5 %. Застосування методу дотренування на автоматично сформованому наборі трійок дало змогу підвищити цей показник до 77,9 %.

Виявлені особливості наявних рішень підтверджують доцільність розроблення власної системи визначення контекстуальних синонімів, орієнтованої на українську мову та побудованої на основі контекстуальних векторних представлень із дотренованою моделлю PMMB2.

РОЗДІЛ 2.
РОЗРОБЛЕННЯ СИСТЕМИ ДЛЯ ДОБОРУ КОНТЕКСТУАЛЬНИХ СИНОНІМІВ

Розроблення системи охоплює низку взаємопов’язаних компонентів. Центральним є сховище контекстуальних векторних представлень лексем, побудоване на основі українськомовних корпусів, що слугує базою для пошуку контекстуальних синонімів. Поруч із ним сформовано сховище векторів із тлумаченнями на основі прикладів із СУМ. Обидва сховища пройшли ряд послідовних етапів очищення, після чого злиті в єдину збірку. На основі неї розгорнуто модуль морфологічного узгодження результатів, компонент вилучення гіперонімів зі словникових дефініцій і вебзастосунок для взаємодії з системою.

Для реалізації обрано мову програмування Python з огляду на широкий набір інструментів для NLP, зокрема, бібліотеки для роботи з трансформерними моделями (transformers, torch), векторного пошуку (FAISS) та мовних аналізаторів.

2.1. Формування збірки корпусних векторів

Для узагальнення на рисунку 2.1 схематично представлено етапи формування збірки ембедингів. Далі в розділі описано логіку створення збірки на кожному з етапів.

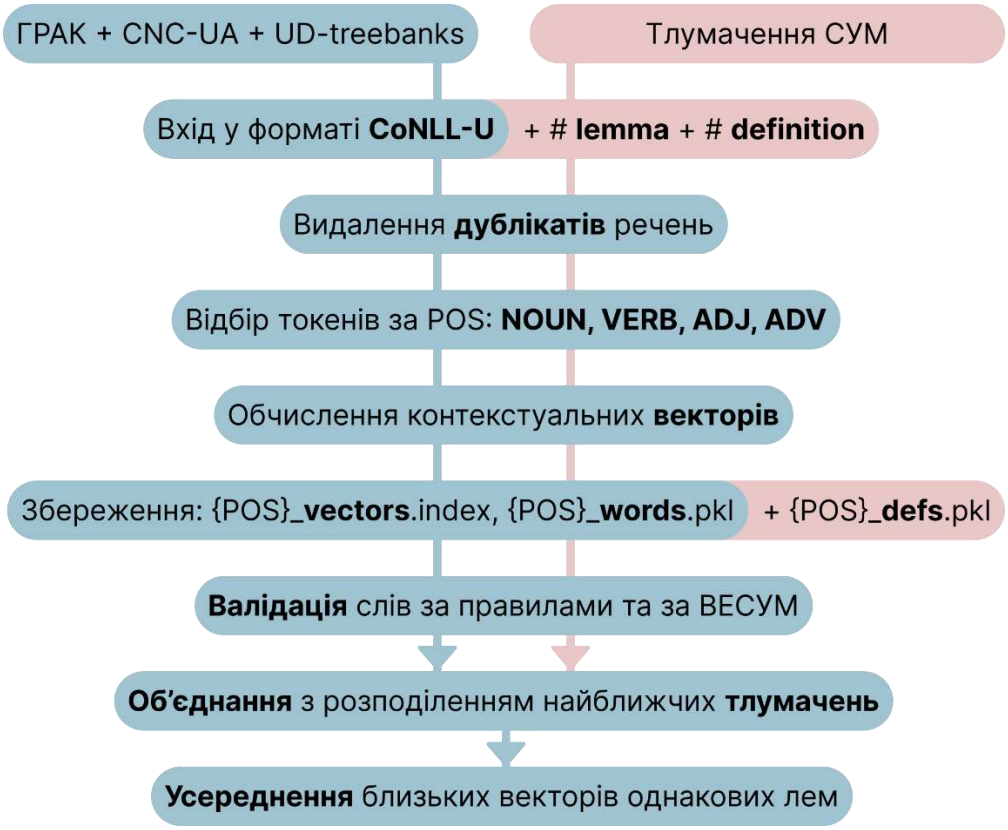

Рисунок 2.1. Етапи формування збірки векторних представлень

2.1.1. Структура вхідних даних і попереднє оброблення корпусів

Вхідними даними для побудови основного сховища векторів є текстові корпуси у форматі CoNLL-U: ГРАК, CNC-UA та UD-treebanks. Цей формат зберігає кожне речення як блок рядків із морфосинтаксичною анотацією: для кожного токена вказано його форму, лему, частину мови, морфологічні ознаки та синтаксичні зв'язки. Речення відокремлені порожнім рядком, а метадані позначені рядками-коментарями, що починаються з «#», зокрема, рядок «# text =» містить вихідний текст речення.

Попередньою умовою для опрацювання CNC-UA стало видалення дублікатів: через особливості формування корпусу одне й те саме речення могло входити в нього кілька разів.

Вхідний CoNLL-U файл опрацьовано генераторною функцією, що повертає речення одне за одним без завантаження всього файлу в пам'ять. Для

кожного речення вона повертає текст і список токенів — кортежів (форма, лема, UPOS).

Обробленню підлягають лише ті токени, UPOS яких належить до чотирьох частин мови: іменник (NOUN), прикметник (ADJ), дієслово (VERB) та прислівник (ADV). Такий вибір зумовлений тим, що слова саме цих частин мови є носіями змістового навантаження і стають об’єктами синонімічної заміни.

2.1.2. Обчислення та зберігання контекстуальних векторів

У розробленому підході запрограмовано обчислення вектора не для словоформи, а для леми слова в контексті речення: перед подаванням до моделі відповідну словоформу замінено лемою, узятою з файлу CoNLL-U. Це забезпечує уніфіковане представлення: усі вектори в сховищі прив’язані до лем, що спрощує подальший пошук і запобігає впливу граматичної форми слова на його вектори.

Обчислення здійснює функція word_embedding, що представлена на рисунку 2.2. Речення обробляє токенізатор моделі PMMB2, побудований на основі SentencePiece, і ділить слова на субтокени, позначаючи початок нового слова символом «▁». Метод get_subtoken_indices знаходить індекси субтокенів цільової леми, послідовно реконструюючи слова із субтокенів і зіставляючи з лемою. Механізм передає речення до моделі та вилучає з її останнього прихованого шару стани субтокенів цільового слова. Їхнє поелементне середнє (mean pooling на рівні слова) утворює фінальний вектор розмірністю 768, який нормалізовано до одиничної довжини для коректного обчислення косинусної подібності через внутрішній добуток.

```python
def word_embedding(sentence, target_word):
    encoded = tokenizer(sentence, return_tensors="pt").to(device)
    try:
        idxs = get_subtoken_indices(sentence, target_word)
    except ValueError:
        return None
    with torch.no_grad():
        outputs = model(**encoded)
        hidden = outputs.last_hidden_state[0]
    vectors = hidden[idxs]
    return vectors.mean(dim=0)
```

Рисунок 2.2. Функція обчислення ембедингу слова в реченні

Для зберігання та пошуку векторів застосовано бібліотеку FAISS [28]. Сховище організоване у вигляді чотирьох пар файлів — по одній на кожну частину мови: FAISS-індекс ({POS}_vectors.index) та pkl-файл із переліком лем ({POS}_words.pkl). Позиція запису в переліку лем відповідає позиції вектора в індексі.

Використано індекс типу IndexFlatIP (плаский індекс на основі внутрішнього добутку), що дає точний, а не наближений результат пошуку. Механізм додає вектори до індексу фрагментами по 200 речень, зберігаючи номер останнього опрацьованого речення у файл, що уможливлює відновлення процесу після переривання.

2.2. Формування збірки з тлумаченнями

2.2.1. Підготування прикладів із СУМ

Окремим джерелом для формування сховища з тлумаченнями слугують речення-приклади із СУМ, підготовлені у форматі CoNLL-U. Кожне таке речення доповнене метаданими: коментарем «# lemma =» із лемою, для якої наведено приклад, та «# definition =» із відповідним словниковим тлумаченням.

Під час оброблення речень із СУМ програма перевіряє відповідність: якщо лема токена збігається з лемою з коментаря «# lemma =», то такий вектор разом із відповідним тлумаченням належатиме окремому сховищу embs_def. Структура embs_def аналогічна основному сховищу, але додатково містить pkl-файли з переліком тлумачень ({POS}_defs.pkl), синхронні з індексами та переліком лем.

2.2.2. Коригування тлумачень

Тлумачення із СУМ потребують автоматичного коригування: у словнику широко вжито типові конструкції, що позначають похідні значення або відсилають до інших статей словника, а не формулюють власне змістове визначення. Виділено чотири групи конструкцій із відповідними правилами оброблення.

Правило 1 (перенаправлення). Якщо тлумачення починається з однієї зі спеціальних конструкцій («Те саме, що», «Жін. до», «Підсил. до», «Збірн. до», «Збільш. до», «Уживається», «Найвищ. ст.»), у ньому міститься покликання на іншу лему. Перше слово після такої конструкції прийнято за цільову лему, а для поточного вектора засобом функції find_closest_def знайдено тлумачення саме цієї леми в embs_def — через пошук семантично найближчого запису за косинусною схожістю векторів, як показано на рисунку 2.3.

```python
def find_closest_def(word, current_vec, lemma_to_entries):
    entries = lemma_to_entries.get(word)
    if not entries:
        return None
    if len(entries) == 1:
        d = entries[0][1]
        return d if is_valid_def(d) else None
    candidate_vecs = np.stack([e[0] for e in entries], axis=0)
    sims = candidate_vecs @ current_vec
    order = np.argsort(sims)[::-1]
    for idx in order:
        d = entries[idx][1]
        if is_valid_def(d):
            return d
    return None
```

Рисунок 2.3. Функція пошуку найближчого тлумачення слова

Правило 2 (найближче власне тлумачення). Якщо тлумачення починається з «Дія за знач.» або «Дія і», механізм шукає запис поточної леми в embs_def, вектор якого є найближчим до вектора поточного запису, підставляючи його тлумачення на місце поточного.

Правило 3 (очищення). Якщо тлумачення починається з конструкцій «Прикм.», «Дієпр.», «Присл.», «Абстр.», «Якість за», «Властивість за», «Стан за знач.» тощо — слово є семантично похідним і власного змістового тлумачення не потребує. Поле тлумачення такого запису залишається порожнім.

Правило 4 (видалення запису). Якщо тлумачення починається з «Перша частина» або «Звуконаслідування», то весь запис видалено зі сховища.

Додатково, якщо тлумачення починається на географічний префікс («У <назва> —»), цей префікс відкинуто перед подальшим обробленням. Після застосування всіх правил сховище embs_def містить коректні тлумачення, придатні для вилучення гіперонімів.

2.3. Очищення сховищ векторів

Після початкового формування обидва сховища — embs та embs_def — проходять незалежне очищення за однаковими правилами.

2.3.1. Фільтрація лем за формальними правилами

На рисунку 2.4. представлено функцію is_valid_lemma, що перевіряє кожну лему. На початку нормалізовано апостроф: усі його варіанти (типографічний, прямий, знак наголосу) замінено на стандартний апостроф (U+02BC).

```python
def is_valid_lemma(lemma):
    normalized = APOSTROPHE_VARIANTS.sub(APOSTROPHE, lemma)
    # довжина
    effective = normalized[3:] if normalized.startswith('по-') else normalized
    if len(effective.replace(APOSTROPHE, '').replace('ь', '')) < 2:
        return False, 'too_short'
    # лише українські
    for ch in effective:
        if ch not in UKRAINIAN_LETTERS and ch != APOSTROPHE:
            return False, f'invalid_char:{ch}'
    # абревіатура
    letters_only = [ch for ch in normalized if ch.isalpha()]
    if letters_only and all(ch.upper() in [l.upper() for l in UKRAINIAN_LETTERS]
                            and ch.isupper() for ch in letters_only):
        return True, normalized
    # велика літера
    if any(ch.isupper() for ch in normalized):
        return False, 'uppercase'
    # апостроф і м'який знак
    return _check_apostrophe_and_soft(normalized)
```

Рисунок 2.4. Функція перевірки леми за формальними правилами

Далі лему перевірено за такими умовами:

- ефективна частина (без апострофа та м’якого знака) містить щонайменше два символи;
- сформована лише з літер українського алфавіту та може містити апостроф;

- не містить великих літер, якщо не є абревіатурою;
- апостроф стоїть лише після губних приголосних і лише перед йотованими голосними (я, ю, є, ї);
- м’який знак стоїть лише після зубних приголосних (д, т, з, с, ц, л, н, р).

Леми, що порушують хоча б одну з умов, видалено разом із відповідними векторами.

2.3.2. Перевірка за ВЕСУМ

Другим кроком є зіставлення відфільтрованих лем зі словником ВЕСУМ у текстовому форматі з розширенням .lst. Кожен рядок цього файлу містить лему та її морфологічну мітку. З файлу вилучено слова з тегами прикметників (/adj), прислівників (adv), дієслів (/v) та іменників (/n) і сформовано словникові множини для кожної частини мови. Додатково відфільтровано слова з частинами числівникових основ (одно-, дво-, три- тощо), адже такі складні слова вказують на точну кількість, що робить їх заміну синонімами недоцільною. Леми зі сховища, які відсутні у відповідній множині ВЕСУМ, вилучено зі сховища.

2.4. Злиття та дедуплікація

Після очищення окремих сховищ здійснено їхнє злиття: для кожної частини мови векторний індекс і переліки лем та тлумачень з embs та embs_def об’єднано в єдиному масиві сховища embs_merged. Вектори з embs, що не мають відповідного тлумачення, отримують None у полі дефініції.

Наступним кроком є дедуплікація: для кожної унікальної леми знайдено всі вектори, що їй відповідають, і серед них об’єднано ті, косинусна подібність між якими перевищує поріг 0,95. Для пошуку зв’язних компонентів подібності застосовано структуру Union-Find [29], що виявляє групи близьких векторів, як зображено на рисунку 2.5.

```
uf = UnionFind(n)
merged_defs = list(defs)
multi_lemmas = {w: idxs for w, idxs in lemma_to_idxs.items()
                if len(idxs) > 1}
merge_count = 0
skipped_count = 0
for lemma, idxs in multi_lemmas.items():
    k = len(idxs)
    sub_vecs = vecs[idxs]
    sim_matrix = sub_vecs @ sub_vecs.T
    for a in range(k):
        for b in range(a + 1, k):
            if sim_matrix[a, b] < sim_thresh:
                continue
            ia, ib = idxs[a], idxs[b]
            ra, rb = uf.find(ia), uf.find(ib)
            if ra == rb:
                continue
            compatible, resolved = defs_compatible(
                merged_defs[ra], merged_defs[rb]
            )
            if not compatible:
                skipped_count += 1
                continue
            uf.union(ra, rb)
            new_root = uf.find(ia)
            merged_defs[new_root] = resolved
            merge_count += 1
```

Рисунок 2.5. Фрагмент коду для дедуплікації записів у збірці

Вектори кожної групи усереднено, а результат нормалізовано. Тлумаченням заведено вважати перше непорожнє значення серед елементів групи. Завдяки дедуплікації усунено надлишкові записи, що виникають через повторювані контексти вживання слова в корпусах.

2.5. Морфологічне узгодження синонімів

Система повертає результати у вигляді лем. Оскільки в природному тексті слова вживають у певній граматичній формі, необхідно поставити запропонований синонім у граматичну форму цільового слова в реченні. Для

трьох підтримуваних частин мови реалізовано окремий механізм, що відповідає особливостям їхньої словозміни. Прислівник є незмінюваною частиною мови, тому леми синонімів метод повертає без змін.

Морфологічні ознаки цільового слова визначено через звернення до мовних аналізаторів UDPipe, Stanza та spaCy. Для вхідного речення та цільового слова аналізатори повертають CoNLL-U-розмітку, з якої програма витягує UPOS і поле Feats із морфологічними ознаками у форматі Universal Features (відмінок, рід, число, особа, час, спосіб тощо). Ці ознаки описують граматичну форму цільового слова та слугують вхідними даними для генерації форми синоніма.

2.5.1. Відмінювання прикметників

Для прикметників реалізовано правилозалежний підхід на основі таблиці флексій. У ній задано закінчення для кожної комбінації відмінка (називний, родовий, давальний, знахідний, орудний, місцевий) і роду/числа (чоловічий, жіночий, середній, множина). На рисунку 2.6 наведено функцію change_adj, що відтинає від леми стандартне двосимвольне закінчення (наприклад, «-ий» або «-ій») і добирає потрібну флексію з таблиці.

```
def change_adj(lem, case, gen):
    base = lem[:-2]
    lem_flex = lem[-2:]
    flex = adj_flexion[case][gen]
    if lem_flex != 'ий':
        if flex[0] in ('и', 'і'):
            flex = lem_flex[0] + flex[1:]
        elif flex[0] == 'о':
            flex = ('ь' if lem_flex[0] == 'і' else 'й') + flex
        elif flex in hard_soft.keys():
            flex = hard_soft[flex]
    return base + flex
```

Рисунок 2.6. Функція відмінювання прикметника

Якщо флексія леми не «-ий» (тобто прикметник не належить до твердої групи), change_adj коригує закінчення:

- якщо флексія починається з «и» або «і», перший символ флексії замінюється на відповідний символ із леми («синім» замість «синим»);
- якщо закнічення починається з «о», до нього додається «й» — якщо флексія леми починається на «ї» («безкрайого» замість «безкраого») — або «ь» — якщо на «-і» («давнього» замість «давного»);
- якщо флексія є однолітерною голосною твердого ряду з таблиці перетворень hard_soft («а»→«я», «у»→«ю», «е»→«є»), її замінено на відповідний м’який варіант («справжню» замість «справжну»).

Такий підхід не потребує зовнішніх ресурсів і працює з будь-яким прикметником-лемою незалежно від типу його основи.

2.5.2. Відмінювання іменників

Для іменників програма отримує форму синоніма через API сервісу vesum.org [30]. Клас VesumAPI реалізує двокроковий запит: спочатку він шукає парадигми слова за його лемою (маршрут POST /grammar/search), отримуючи ідентифікатор парадигми, а потім за цим ідентифікатором запитує всі словоформи парадигми (маршрут GET /grammar/details/{pdg_id}). З отриманого переліку програма відбирає ту форму, тег якої відповідає потрібній комбінації відмінка та числа.

Для перетворення ознак із формату Universal Features у теги vesum.org застосовано відповідність: перший символ назви відмінка і перший символ назви числа об’єднано у двобуквений тег (наприклад, «Gen» + «Sing» → «GS», «Dat» + «Plur» → «DP»). Реалізовано повернення всіх варіантів для заданої форми. Це враховує наявність паралельних форм у деяких іменників.

2.5.3. Дієвідмінювання

Для дієслів використано файл, попередньо побудований зі словникового файлу ВЕСУМ. Клас VerbFormsCache завантажує цей файл у пам’ять на ініціалізації.

Під час побудови кешу метод опрацьовує всі рядки словникового файлу: для кожної пари (словоформа, лема) з тегом дієслова програма розкладає тег для визначення часу, способу дії, особи, числа та роду. Леми нормалізовано: відкинуто рефлексивний суфікс (-ся/-сь) й інфінітивне закінчення (-ти/-ть). Для нормалізованої леми збережено кортеж із 19-ти позицій, що відповідають усім відмінюваним формам: теперішній (6 форм), минулий (4 форми), майбутній (6 форм) час і наказовий спосіб (3 форми).

Після запиту програма нормалізує лему, визначає індекс потрібної форми та повертає її зі збереженого кортежу. Для зворотних дієслів приєднує суфікс «-ся» з урахуванням правила: у формі 3-ї особи однини теперішнього та майбутнього часу, де основа закінчується на «е» або «є», суфіксом стає «-ться», а не «-ся».

2.6. Вилучення гіперонімів із тлумачень

Тлумачення в СУМ побудовано за принципом поєднання родового слова та видової відмінності: перше значуще слово або словосполучення позначає родове поняття — гіперонім — до тлумаченого слова. Наприклад, дефініція іменника «аврал» — «Виконувана всім колективом спішна робота» — містить гіперонім «робота», виражений іменником, тобто тією частиною мови, що й слово «аврал». Вилучення гіперонімів із тлумачень розширює результати системи: користувач може отримати узагальнювальне поняття, придатне для змістового перефразування замість лексичної заміни.

Розроблений механізм вилучення гіперонімів ґрунтується на синтаксичному аналізі тексту тлумачення засобами мовних аналізаторів. Для кожного запису з непорожнім тлумаченням у сховищі embs_def програма отримує CoNLL-U-розмітку. Якщо тлумачення містить кілька субдефініцій, розділених крапкою з комою, то механізм передає кожну субдефініцію до мовних аналізаторів і обробляє їх незалежно, оскільки синтаксичний розбір тексту з крапкою з комою (;) всередині речення може бути сприйнятим аналізаторами як окремі речення.

Пошук гіперонімів у CoNLL-U-розмітці реалізовано на основі списку токенів одного речення та цільової частину мови (UPOS) слова, для якого записано тлумачення. Розроблений механізм шукає гіпероніми в такій послідовності:

1. Визначення кореня речення. З переліку токенів вилучено вузол із синтаксичним зв’язком deprel = root.

2. Перевірка кореня. Якщо UPOS кореня збігається з цільовою частиною мови, корінь є якорем-гіперонімом. Ця ситуація є типовою: наприклад, у тлумаченні іменника «зошит для записів, нотаток» іменник «зошит» є коренем речення.

3. Пошук кандидата серед нащадків кореня. Якщо UPOS кореня не збігається з цільовою частиною мови (така ситуація виникає, коли тлумачення починається з дієприкметника або прикметника), метод шукає найближчий вузол із потрібним UPOS серед нащадків кореня. Початково увагу приділено прямим нащадкам кореня, відсортованим за пріоритетом синтаксичного зв’язку: nsubj (граматичний підмет) має найвищий пріоритет, далі — obj, xcomp, ccomp, obl, nmod. Якщо серед прямих нащадків потрібного вузла не виявлено, програма шукає його по решті дерева залежностей.

4. Збирання однорідних членів. Від знайденого гіпероніма функція _collect_conj рекурсивно збирає всі вузли, пов’язані з ним через зв’язок deprel = conj і тим самим UPOS. Це охоплює як пласкі структури (усі однорідні члени

прив’язані безпосередньо до кореня), так і ланцюгові (A → conj B → conj C). Так функція знаходить усі однорідні гіпероніми в тлумаченнях на зразок «книжка або зошит для записів, нотаток» (книжка, зошит).

Отримані ідентифікатори токенів-гіперонімів програма перетворює на символьні діапазони за полем TokenRange з misc-стовпця CoNLL-U. Поле TokenRange задає точні символьні позиції токена у вихідному рядку. Далі система вставляє квадратні дужки до текст тлумачення за відсортованими символьними діапазонами без зміни решти тексту. Результатом є тлумачення, де гіпероніми виокремлені квадратними дужками, наприклад: «[книжка] або [зошит] для записів, нотаток».

Оброблені тлумачення програма зберігає в новому сховищі: оновлений pkl-файл дефініцій записує атомарно через тимчасовий файл, щоб унеможливити пошкодження даних у разі перериванні процесу. Векторний індекс і перелік лем не зазнають змін.

2.7. Пошук контекстуальних синонімів

Центральний механізм системи реалізовано в методі, що приймає речення з цільовим словом і повертає впорядкований список кандидатів-синонімів із показниками семантичної подібності, а також найближче тлумачення слова з виділеними гіперонімами, якщо таке наявне. На рисунку 2.7 узагальнено логіку добору контекстуальних синонімів.

Рисунок 2.7. Схема добору контекстуальних синонімів

Механізм сформовано з таких кроків:

1. Морфологічний аналіз. Через мовні аналізатори для вхідного речення та цільового слова система визначає лему та UPOS, а також морфологічні ознаки (відмінок, рід, число, особа тощо).
2. Обчислення вектора запиту. Механізм замінює словоформу лемою, після чого метод word_embedding обчислює нормалізований контекстуальний вектор.
3. Векторний пошук. У FAISS-індексі відповідної частини мови програма шукає top-k найближчих векторів за внутрішнім добутком, що еквівалентний косинусній подібності для нормалізованих векторів.
4. Постфільтрація. Із результатів вилучено лему цільового слова; кандидатів, чия подібність нижча за встановлений поріг, відкинуто.
5. Морфологічне узгодження. Для кожного відібраного кандидата визначено правильну словоформу відповідно до ознак цільового слова.
6. Долучення тлумачення з гіперонімами. Якщо для цільового слова в сховищі наявні записи з тлумаченнями, програма обчислює його вектор і

шукає у FAISS-індексі. Серед знайдених записів відбирається той, косинусна подібність якого до вектора є найвищою. Відповідне тлумачення разом із виділеними в ньому гіперонімами програма долучає до відповіді.

2.8. Удосконалення системи за результатами оцінювання

Під час оцінювання роботи системи, описаного в підрозділі 3.1.4, виявлено два характерні недоліки. По-перше, обчислення вектора цільового слова на основі повного речення призводило до того, що семантично нерелевантні слова речення, синтаксично не пов’язані з цільовим словом, вносили шум у його векторне представлення. По-друге, ранжування кандидатів за подібністю між векторами двох слів допускало контекстуальну невідповідність: слово може бути семантично близьким до цільового, але після підставлення в речення змінювати його загальний зміст. Для усунення цих недоліків розроблено два вдосконалення.

2.8.1. Формування контексту з урахуванням синтаксису

Перше вдосконалення стосується способу побудови вхідного рядка для моделі перед обчисленням вектора цільового слова. Замість повного тексту речення модель отримує синтетичний контекст, де цільове слово розташоване в центрі, а його оточення утворено лише синтаксично пов’язаними з ним словами.

Для формування такого контексту програма використовує синтаксичне дерево залежностей речення, побудоване на основі роботи мовних аналізаторів. Із дерева механізм вилучає два набори вузлів: предки цільового слова (вузли, від яких воно залежить) та нащадки (вузли, що залежать від цільового слова, рекурсивно). Обидва набори обмежені трьома рівнями зв’язку — тобто до сховища потрапляють лише вузли, що перебувають не далі ніж на трьох кроках

вгору або вниз по дереву від цільового слова, без урахування службових частин мови.

Модель отримує підготовлений контекст у форматі «[предки] [цільове слово] [нащадки]». Таке впорядкування імітує природну валентність слова: предки задають семантичний клас вживання, а нащадки уточнюють або розширюють його в межах поточного контексту.

Обґрунтуванням такого підходу є властивості семантики: значення слова визначено його найближчим синтаксичним оточенням сильніше, ніж словами, синтаксично від нього відокремленими. Повне речення містить елементи, що описують паралельні або підрядні конструкції, семантично не пов’язані з цільовим словом. Їхнє вилучення дає змогу зосередити увагу моделі саме на тих словах, що безпосередньо визначають значення цільового слова в певному вживанні.

Додатково в межах першого вдосконалення виконано статичну квантизацію моделі — перетворення вагових параметрів із вищої числової точності на нижчу — з метою зменшення витрат пам’яті, а також для прискорення обчислень під час формування збірки.

Для вибору оптимального формату квантизації порівняно три варіанти: базовий float32, float16 та int8-квантизація засобами ONNX Runtime [31]. Порівняння здійснено в середовищі Google Colab [32] з графічним процесором T4 за трьома критеріями: точністю WSD на оцінювальному наборі з дослідження [4], розміром моделі на диску та часом оброблення набору моделлю. Результати наведено в таблиці 2.1.

Таблиця 2.1. Порівняння варіантів квантизації моделі

Критерій	**float32**	**float16**	**int8**
Точність WSD, %	77,91	77,87	76,73
Розмір на диску, МБ	1081,8	551,5	265,8
Час оброблення, с	20,16	4,42	65,33

З таблиці 2.1 видно, що квантизація до float16 зменшує розмір моделі в 1,96 раза та прискорює оброблення тестового набору в 4,56 раза порівняно з початковим float32, втрачаючи лише 0,04 % точності. Натомість варіант з int8, попри найменший розмір на диску, помітно поступається за часом оброблення.

Квантизацію до float16 виконано через перетворення всіх вагових параметрів моделі з 32-бітного формату з рухомою комою до 16-бітного. Для int8-квантизації попередньо виконано калібрацію на текстах з оцінювального набору, щоб отримати статистику розподілу активацій. На її основі для кожного шару моделі обчислено параметри масштабування та нульової точки. З'ясовано, що квантизація до int8 є непрактичною з огляду на тривалий час оброблення текстів моделлю. Отож для формування збірки обрано формат float16 як компроміс між точністю, розміром і швидкістю.

2.8.2. Переранжування кандидатів на рівні речення

Перше вдосконалення покращує якість вектора цільового слова, однак ранжування кандидатів-синонімів усе ще спирається на подібність між двома словесними векторами, а не на реальну семантичну сумісність кандидата з реченням. Для усунення цього обмеження розроблено другий метод — переранжування на рівні цілого речення.

Отримавши список кандидатів (відповідно до способу, описаного в підрозділі 2.7), удосконалений механізм підставляє їх у вхідне речення на місце цільової словоформи, утворюючи нове речення. Для кожного такого речення-кандидата обчислено векторне представлення на рівні речення: модель PMMB2 отримує все речення, і вектор сформовано як mean pooling прихованих станів усіх його токенів — тобто як стандартний ембединг речення [23, с. 3984]. Аналогічно отримано вектор початкового речення. Показник близькості

кандидата замінено на косинусну подібність між векторами вхідного та новоствореного речень.

Логіка такого підходу ґрунтується на тому, що справжній контекстуальний синонім при підставлянні в речення має зберігати його загальний семантичний зміст, тобто ембединг речення з синонімом повинен бути максимально близьким до вектора вхідного речення. Так ранжування більше не залежить від якості окремих словесних векторів і переходить до прямого порівняння смислів речень, що є надійнішим критерієм семантичної рівнозначності заміни з огляду на результати оцінювання в підрозділі 3.1.4. Кандидати, що спотворюють загальний зміст речення попри словесну близькість до цільового слова, отримують нижчу оцінку та позицію у вислідному списку.

2.9. Створення вебзастосунку

2.9.1. Серверна частина

Серверну частину реалізовано за допомогою фреймворку FastAPI [33] на Python. FastAPI забезпечує автоматичну валідацію вхідних даних. Програма завантажує модель PMMB2 і FAISS-індекси один раз під час запуску сервера та зберігає їх у пам’яті впродовж усього сеансу роботи, запобігаючи їхньому повторному завантаженню на кожному запиті.

Реалізовано маршрут POST «/synonyms», що приймає JSON-тіло з полями: sentence (речення з цільовим словом), word (слово у будь-якій граматичній формі), а також необов’язкові min_similarity (мінімальний поріг косинусної подібності) та max_nearest (ліміт кількості результатів). Сервер викликає функцію добору синонімів із морфологічним узгодженням знайдених слів і повертає відповідь із такими полями: word і lemma_used (вихідне слово та його лема), pos (частина мови: NOUN, VERB, ADJ або ADV), feats (морфологічні ознаки), query_definition (тлумачення цільового слова, якщо доступне), results (список семантично близьких слів у тій самій граматичній формі, що й слово-

запит, із значеннями косинусної подібності) та total (кількість знайдених результатів). Приклад запиту та відповіді сервера наведено в додатку А.

2.9.2. Клієнтська частина

Клієнтську частину реалізовано у вигляді односторінкового застосунку з використанням бібліотеки React [34]. Інтерфейс складається з двох основних частин: поля введення речення з виокремленням цільового слова і панелі результатів із переліком знайдених синонімів і гіперонімів.

При виокремленні слова в реченні клієнт надсилає POST-запит до сервера та відображає результат у вигляді інтерактивного списку. Для кожного синоніма виведено показник косинусної подібності. Демонстрацію роботи вебзастосунку наведено в підрозділі 3.2.2.

2.10. Висновки до розділу 2

У другому розділі кваліфікаційної роботи описано розроблення NLP-системи для добору контекстуальних синонімів в українськомовних текстах.

Сформовано два незалежні сховища векторних представлень. Основне — на матеріалі корпусів ГРАК, CNC-UA та UD-treebanks — містить контекстуальні ембедінги лексем чотирьох частин мови, обчислені через mean pooling прихованих станів субтокенів цільової леми в реченні з нормалізацією до одиничної довжини. Додаткове сховище embs_def сформовано на основі речень-прикладів зі СУМ із пов’язаними тлумаченнями та скориговано за правилами оброблення відсилальних і похідних конструкцій. Обидва сховища пройшли очищення за формальними правилами перевірки лем і фільтрацію за словником ВЕСУМ, після чого об’єднані в єдину збірку. Дедуплікацію виконано методом Union-Find з усередненням груп близьких векторів.

Реалізовано модуль морфологічного узгодження синонімів із граматичним контекстом речення: правилозалежний підхід на основі таблиці флексій для прикметників, звернення до API vesum.org для іменників і попередньо побудований кеш зі словника ВЕСУМ для дієслів.

Розроблено механізм вилучення гіперонімів із тлумачень СУМ засобами морфосинтаксичного аналізу: пошук у дереві залежностей вузла з потрібною частиною мови та збирання його однорідних членів.

Реалізовано центральний механізм пошуку контекстуальних синонімів із лематизацією, векторним пошуком у FAISS і морфологічним узгодженням результатів. Для підвищення якості системи розроблено два вдосконалення: формування синтаксично орієнтованого контексту для обчислення вектора цільового слова з квантизацією моделі до формату float16, а також переранжування кандидатів через порівняння ембедингів речень.

Усі компоненти системи інтегровано у вебзастосунок із серверною частиною на FastAPI та клієнтською на React.

РОЗДІЛ 3.
АПРОБАЦІЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Розділ присвячено оцінюванню якості роботи розробленої системи та демонстрації її практичного застосування. У підрозділі 3.1 описано методику оцінювання та наведено результати тестування на трьох етапах: до вдосконалень, після першого вдосконалення та після обох. Для кожного етапу обчислено точність і повноту на основі порівняння з еталонним словником синонімів, а отримані показники проаналізовано в розрізі чотирьох підтримуваних частин мови. У підрозділі 3.2 продемонстровано отримані результати: описано й унаочнено вміст сформованої збірки ембедингів, наведено приклад роботи системи через інтерфейс вебзастосунку.

3.1. Оцінювання роботи системи на основі синонімічного словника

Для оцінювання якості роботи системи як еталонний ресурс використано словник синонімів [13]. На його основі для кожного аналізованого слова розроблений механізм формує дві множини, а визначені на них два порогові значення косинусної подібності — нижнє і верхнє — слугують основою для обчислення точності та повноти відповідно.

3.1.1. Формування еталонної та кандидатської множин

Нехай w_0 — аналізоване слово, $\{D_1, D_2, \dots, D_n\}$ — усі групи синонімів зі словника, до кожної з яких входить w_0. Оскільки одне слово може належати до кількох синонімічних груп, еталонну множину $S(w_0)$ визначено як об’єднання всіх таких груп без самого слова w_0:

$$S(w_0) = \left(\bigcup_{i=1}^{n} D_i \right) \setminus \{w_0\}.$$

Кандидатську множину $C(w_0)$ сформовано зі збірки контекстуальних ембедингів: до неї входять усі слова, що програма повертає як кандидати-синоніми для w_0. Визначено відображення $f : C(w_0) \to \mathbb{R}$, що ставить у відповідність кожному кандидату w значення косинусної подібності між його вектором і вектором слова w_0. Якщо слово w представлено в збірці кількома векторами (у різних контекстах вживання), то значення $f(w)$ визначено як максимум серед відповідних подібностей:

$$f(w) = \max_{v \in V(w)} cos\_sim(v, e_{w_0}),$$

де $V(w)$ — множина всіх векторів слова w в збірці, e_{w_0}— вектор цільового слова. Вибір максимуму замість середнього зумовлений відповідністю реальній поведінці системи: під час пошуку кандидатів за кількома векторами одного слова програма обирає найближчий до вектора запиту, тоді як середнє штучно занижувало б подібність через контексти, де слово вжите в іншому значенні.

Оскільки всі слова словника синонімів наявні в збірці, дійсним є входження: $S(w_0) \subseteq C(w_0)$.

3.1.2. Визначення нижнього та верхнього порогів подібності

Нижній поріг $\tau_{\text{low}}(w_0)$ визначено як мінімальне значення f серед усіх елементів кандидатської множини:

$$\tau_{\text{low}}(w_0) = \min \{f(w) : w \in C(w_0)\}.$$

Водночас розроблена система формує множину кандидатів так, що вказаний мінімум завжди досягнуто на слові, що належить еталонній множині:

$$\exists w \in S(w_0) : f(w) = \tau_{low}(w_0),$$
$$\forall w \in C(w_0) \setminus S(w_0) : f(w) > \tau_{low}(w_0).$$

Верхній поріг $\tau_{\text{up}}(w_0)$ визначено як найменше значення f, досягнуте на деякому елементі кандидатської множини та починаючи з якого всі кандидати з не меншою подібністю є елементами еталонної множини:

$$\tau_{\text{up}}(w_0) = \min \{f(w) : w \in C(w_0), \forall w' \in C(w_0) : f(w') \geq f(w) \Rightarrow w' \in S(w_0)\}.$$

Еквівалентне формулювання через пряму умову на проміжок:

$$\forall w \in C(w_0) : f(w) \geq \tau_{\text{up}}(w_0) \Rightarrow w \in S(w_0).$$

У проміжку $[\tau_{\text{up}}(w_0), 1)$ розташовані лише елементи еталонної множини — значення 1 відповідає самому w_0, яке до $C(w_0)$ не входить. Натомість у зоні $[\tau_{\text{low}}(w_0), \tau_{\text{up}}(w_0))$ можуть бути наявні слова як з $S(w_0)$, так і з-поза неї, проте слово з подібністю $\tau_{\text{low}}(w_0)$ гарантовано належить еталонній множині.

3.1.3. Обчислення точності та повноти

Точність на нижньому порозі $P(w_0)$ характеризує частку словникових синонімів серед усіх кандидатів із кандидатської множини $C(w_0)$:

$$P(w_0) = \frac{|S(w_0)|}{|C(w_0)|}.$$

Оскільки $S(w_0) \subseteq C(w_0)$, ця величина є нижньою межею точності в системі: будь-яке підвищення порогу відсікання може тільки збільшити або зберегти точність.

Повнота на верхньому порозі $R(w_0)$ характеризує частку синонімів з кандидатської множини, що мають косинусну подібність не нижчу за $\tau_{\text{up}}(w_0)$ і, отже, входять до еталонної множини:

$$R(w_0) = \frac{\left|\{w \in S(w_0) : f(w) \geq \tau_{\text{up}}(w_0)\}\right|}{|S(w_0)|}.$$

Ця метрика відображає, яка частка множини словникових синонімів потрапляє до ділянки, де всі кандидати є такими синонімами.

Усі обчислені величини — τ_{low}, τ_{up}, P та R — усереднено для слів кожної частини мови та для збірки загалом. Нехай W — множина всіх аналізованих слів, тоді:

$$P_W = \frac{1}{|W|} \sum_{w_0 \in W} P(w_0);$$

$$R_W = \frac{1}{|W|} \sum_{w_0 \in W} R(w_0).$$

Пороги усереднено за тією ж схемою. Аналогічно відбувається усереднення в межах слів кожної частини мови.

3.1.4. Оцінки роботи системи з урахуванням удосконалень

Після розроблення початкової версії системи проведено оцінювання якості її роботи. Отримані результати наведено нижче в таблиці 3.1. Для наочності всі

числові показники перетворено у відсоткові значення з точністю до 2-х знаків після коми.

Таблиця 3.1. Оцінка роботи системи до вдосконалень

Група слів	**Пороги та метрики, %**			
	Поріг τ_{up}	**Повнота *R***	**Поріг τ_{low}**	**Точність *P***
Дієслова	95,32	17,14	61,40	32,08
Іменники	96,71	18,29	53,96	30,29
Прикметники	96,35	25,63	57,10	34,78
Прислівники	94,80	23,47	67,12	35,33
Усі	95,80	21,13	59,90	33,12

З таблиці 3.1 видно, що для слів усієї сформованої збірки усереднені значення верхнього та нижнього порогів косинусної подібності дорівнюють 95,80 % і 59,90 % відповідно. Значення повноти на верхньому порозі становить 21,13 %, а точності на нижньому — 34,62 %. Для підвищення якості роботи системи запропоновано 2 способи її вдосконалення, описані в підрозділі 2.8.

Після вдосконалення системи, описаного в підрозділі 2.8.1, знову оцінено якість її роботи.

Таблиця 3.2. Оцінка роботи системи після першого вдосконалення

Група слів	**Пороги та метрики, %**			
	Поріг τ_{up}	**Повнота *R***	**Поріг τ_{low}**	**Точність *P***
Дієслова	97,08	22,37	60,54	35,46
Іменники	95,61	21,48	65,14	38,11
Прикметники	97,10	28,83	61,77	47,30
Прислівники	95,34	27,02	73,36	40,09
Усі	96,28	24,93	65,20	40,24

З результатів у таблицях 3.1 і 3.2 виявлено, що в порівнянні з початковою оцінкою верхній поріг подібності зріс на 0,48 %, а нижній — на 5,3 %. Водночас зросли значення повноти та точності: на 3,8 % і 7,12 % відповідно.

З огляду на отримані результати, запропоновано друге вдосконалення системи, описане в підрозділі 2.8.2 та результати якого наведено нижче.

Таблиця 3.3. Оцінка роботи системи після двох удосконалень

Група слів	**Пороги та метрики, %**			
	Поріг τ_{up}	**Повнота *R***	**Поріг τ_{low}**	**Точність *P***
Дієслова	98,36	45,37	85,25	65,34
Іменники	98,40	46,98	85,51	65,60
Прикметники	98,32	53,09	87,00	73,83
Прислівники	98,21	51,28	86,67	68,35
Усі	98,32	49,18	86,12	68,28

Після порівняння результатів з таблиць 3.1 і 3.3 з’ясовано, що після обох удосконалень верхній поріг косинусної подібності піднявся на 2,52 %, а нижній — на 26,22 %. Більш ніж удвічі зросли значення повноти та точності: на 28,05 % (у 2,06 раза) і на 35,16 % (у 2,33 раза) відповідно. Для унаочнення на рисунку 3.1 зображено графік, де для кожного з етапів розроблення представлено значення порогів і відповідних метрик. Етап I відповідає стану системи до вдосконалень, етап III — після обох удосконалень.

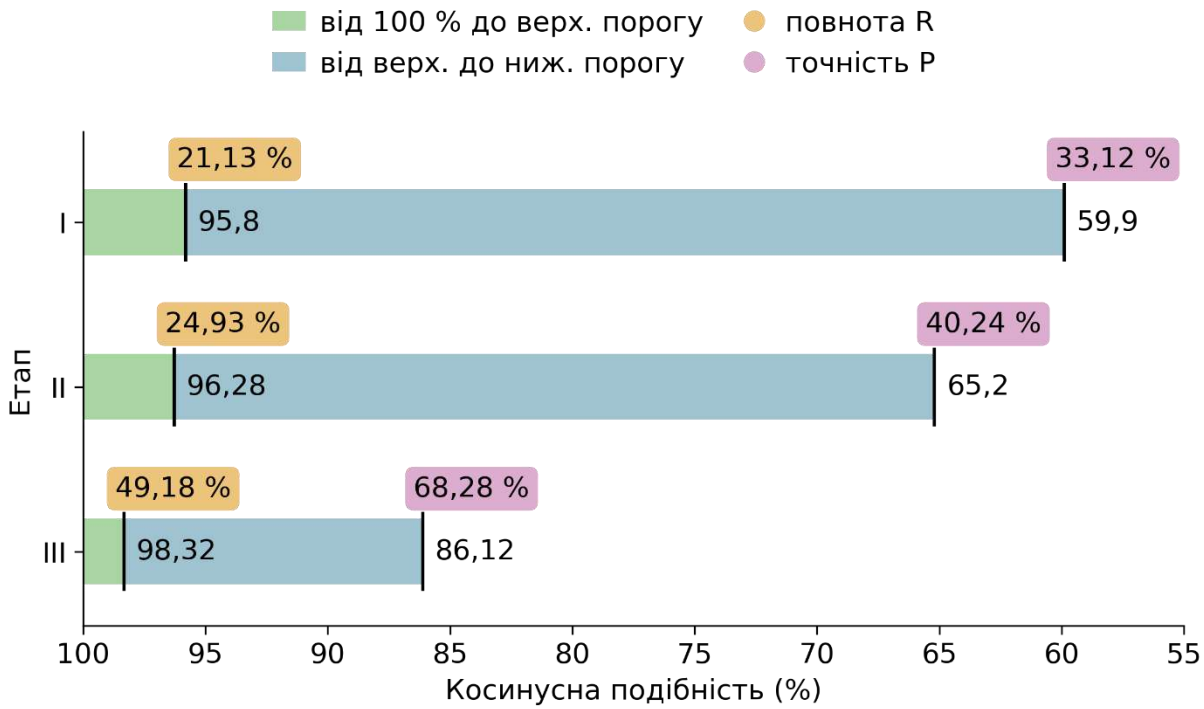

Рисунок 3.1. Оцінки роботи системи на кожному етапі розроблення

3.2. Демонстрація отриманих результатів

У цьому підрозділі представлено й описано візуалізації, що демонструють вміст сформованої збірки ембедингів. Для створення зображень використано Python-бібліотеки Matplotlib [35] і Vega-Altair [36]. Далі продемонстровано роботу розробленого вебзастосунку за допомогою знімків, опису та пояснення функціональності інтерфейсу.

3.2.1. Опис сформованої збірки ембедингів

Сформована збірка векторних представлень містить загалом 1 589 593 записи, серед яких наявні 94 604 унікальні леми. Звідси випливає, що для однієї леми в збірці вміщено в середньому 17 ембедингів. Нижче на рисунку 3.2 унаочнено розподіл кількостей лем за частинами мови. З діаграми видно, що найбільша кількість припадає на прикметники — 674 963 записи, що відповідає

42,46 % від загального розміру збірки. Утім, більш ніж половина (54,13 %) унікальних лем є іменниками: 51 208 слів.

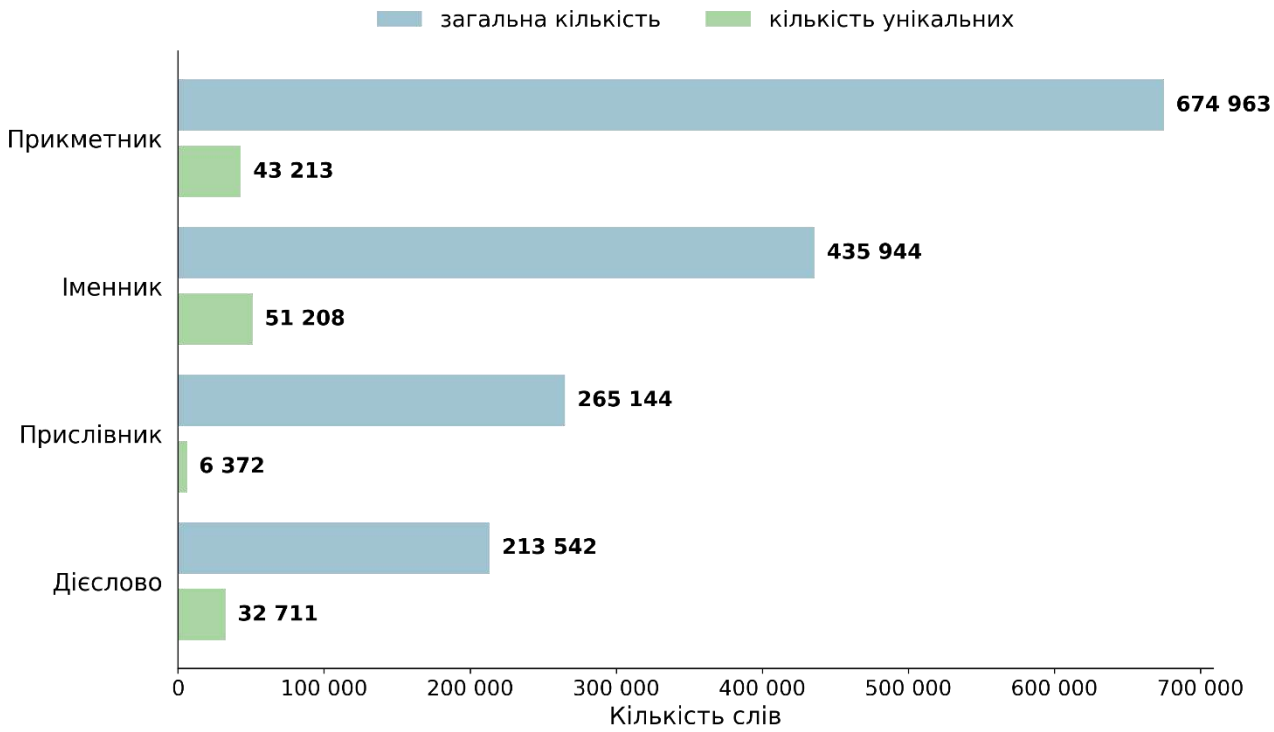

Рисунок 3.2. Графік розподілу кількостей слів за частинами мови

На рисунку 3.3 наведено перші 25 слів за кількістю входжень у збірці. Можна бачити, що на графіку переважають прикметники, тоді як серед дієслів є лише одна лема, а серед іменників — дві. Словом з найбільшою кількістю входжень є прикметник «великий», за ним — «новий». Третім на графіку розташований прислівник «можна» — з відривом у майже 1,5 тис. входжень.

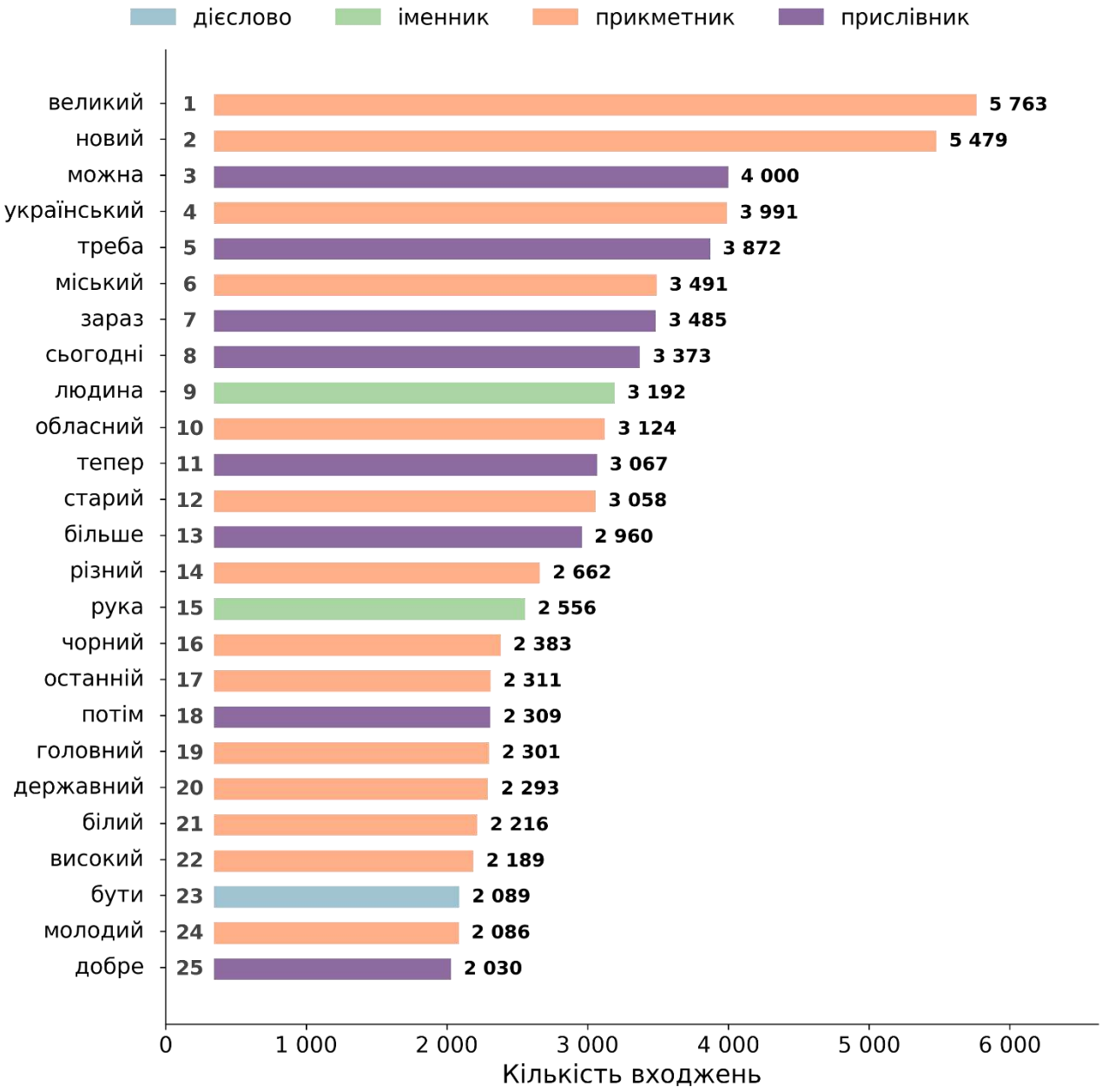

Рисунок 3.3. Розподіл 25-ти найчастотніших слів із частинами мови

Аби унаочнити вміст сформованої збірки, створено проєкцію векторного простору її ембедингів на площину за допомогою UMAP [37]. Як видно на рисунку 3.4, зі збірки взято вектори дієслів: «йти», «рухатися», «бігти», «стояти», «сидіти» та «лежати».

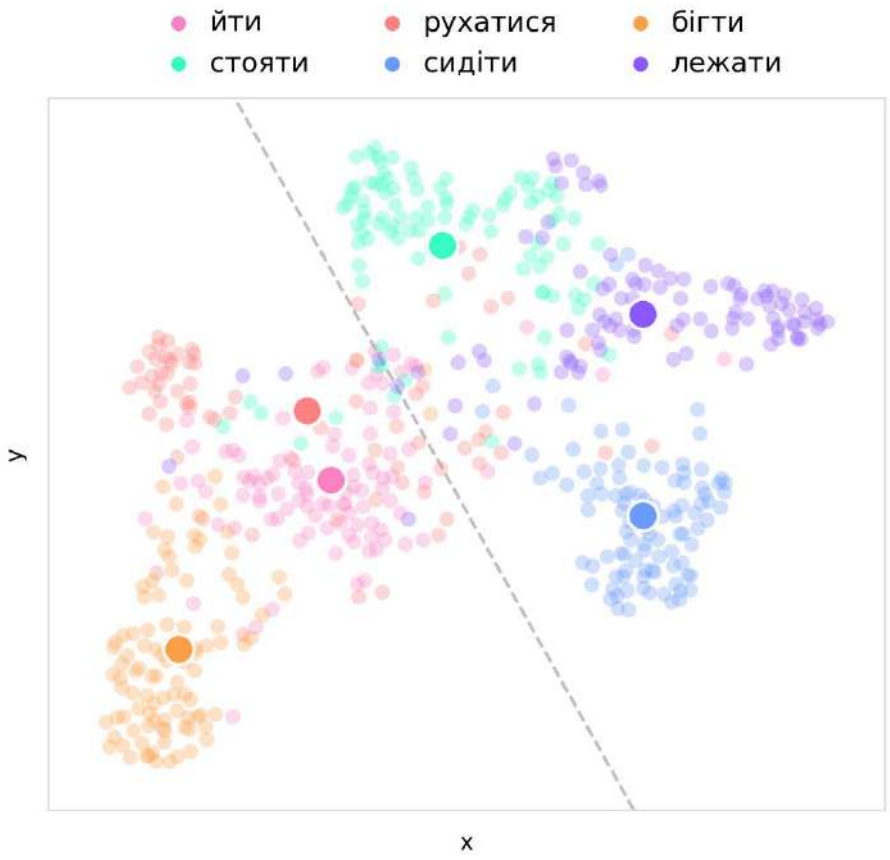

Рисунок 3.4. UMAP-проєкція ембедингів на площину

Ці дієслова утворюють два семантичні кластери: групу слів, що позначають переміщення в просторі («йти», «рухатися», «бігти»), та групу дієслів стану або статичного положення («стояти», «сидіти», «лежати»). На графіку кожне слово відображено як хмару напівпрозорих точок, які репрезентують його окремі контекстуальні входження. Більшими непрозорими маркерами позначено їхні центроїди — усереднені векторні представлення лексем.

Для демонстрації розрізнення між цими поняттями на площину нанесено сіру пунктирну пряму. Вона побудована перпендикулярно до вектора, що з’єднує загальні центри мас обох виділених груп. Сформована візуалізація засвідчує, що вектори семантично подібних слів у досліджуваному просторі згруповані ближче один до одного, а семантично віддалені чи протилежні (динаміка проти статики) — розміщені відокремлено.

3.2.2. Демонстрація роботи вебзастосунку

Як описано в підрозділі 2.9.2, інтерфейс застосунку розроблено у форматі односторінкового сайту. Початково, як показано на рисунку 3.5, ліворуч на сторінці розташовано блок для введення тексту, а праворуч — блок із кнопкою для пошуку синонімів, яка є неактивною, доки користувач не введе текст і не обере слово.

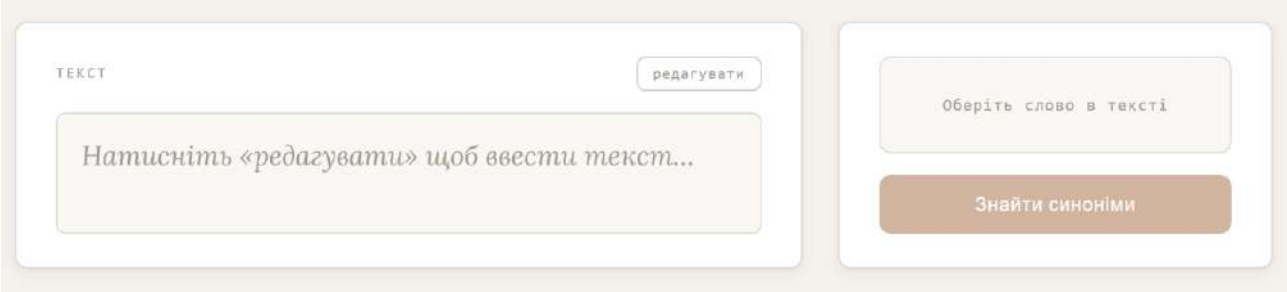

Рисунок 3.5. Інтерфейс вебзастосунку до введення тексту

Для введення тексту користувачу необхідно натиснути кнопку «редагувати», надрукувати або вставити текст і натиснути кнопку «готово», як унаочнено на рисунку 3.6.

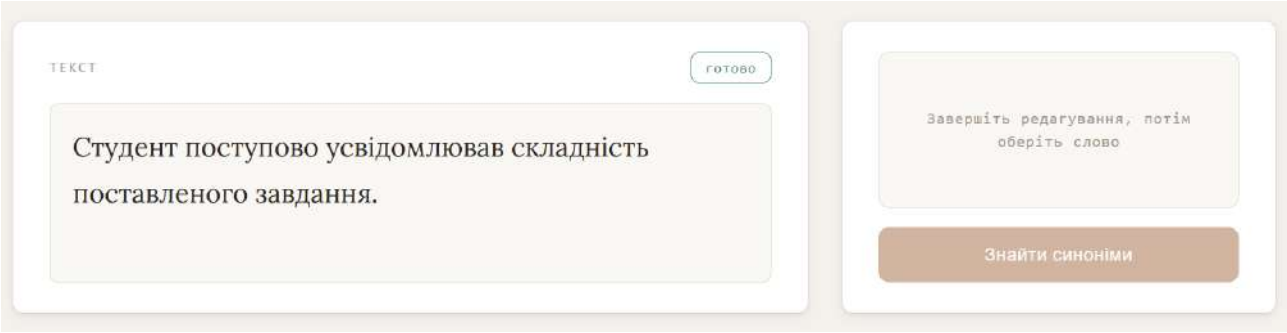

Рисунок 3.6. Інтерфейс після введення тексту

Якщо текст уведено, користувач має змогу натиснути на слово, для якого хоче дібрати контекстуальний синонім. Обране слово стане виділеним і підкресленим у тексті, а також доданим у блок праворуч, як представлено нижче

на рисунку 3.7 зі словом «усвідомлював». Тепер користувач має змогу знайти синоніми.

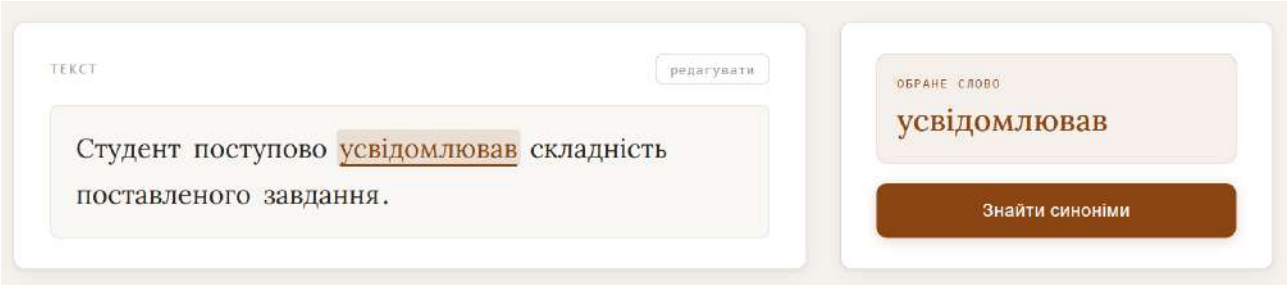

Рисунок 3.7. Інтерфейс після натискання на слово в тексті

Як видно на рисунку 3.8, після натискання на кнопку «Знайти синоніми» на сторінці з’являються ще 2 блоки. Синоніми, знайдені у сформованій збірці, наведено в лівому блоці в порядку спадання подібності. У правому блоці розташовано словникові тлумачення з гіперонімами, обрамленими квадратними дужками. Слід зауважити, що всі знайдені відповідники наведено в тій граматичній формі, що й початкове слово.

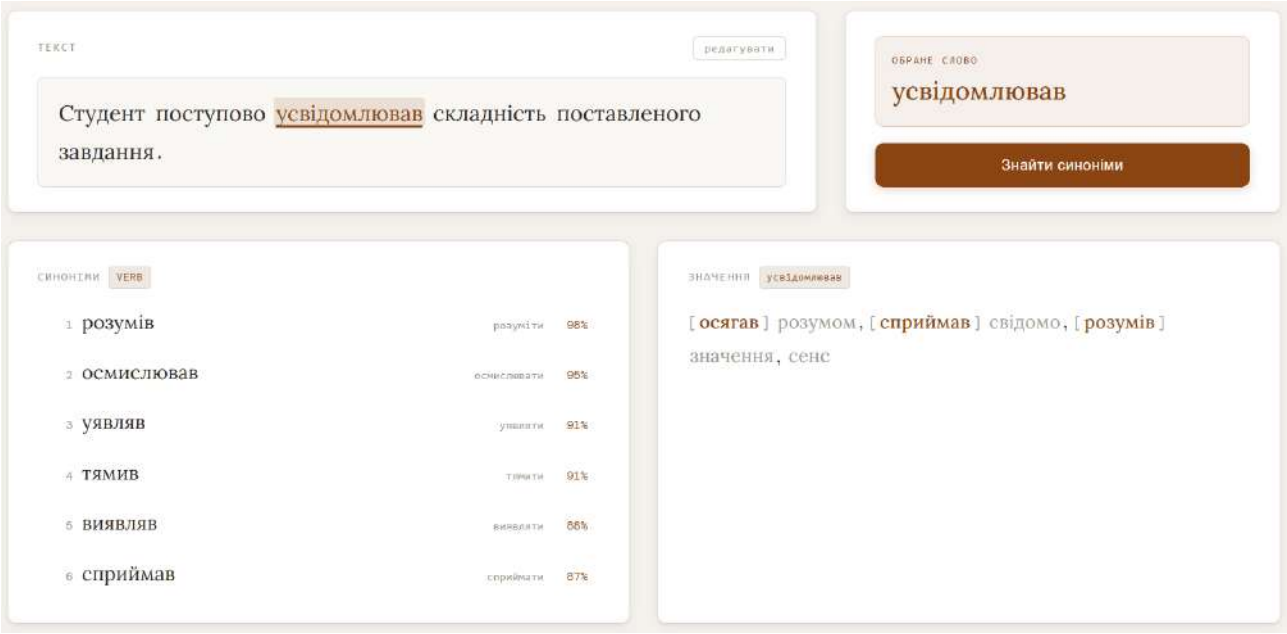

Рисунок 3.8. Інтерфейс зі знайденими синонімами

Якщо користувач натисне на слово з блоку синонімів, програма підставить його замість цільового слова в тексті. Те саме стосується гіперонімів із блоку значень слова. За потреби користувач додаватиме до них пов’язані слова з тлумачення, як зображено на рисунку 3.9 із гіперонімом «осягав» і залежним словом «розумом». На рисунку 3.8 видно, що залежні від гіперонімів слова мають сірий колір шрифту, на відміну від рисунка 3.9, де колір змінився на чорний: користувач не може обрати другорядні слова в тлумаченні, поки не обере гіперонім.

Рисунок 3.9. Інтерфейс із обраними замінниками цільового слова

3.3. Висновки до розділу 3

У третьому розділі комплексно оцінено якість розробленої системи та продемонстровано її практичне застосування.

Запропоновано методику оцінювання на основі словника синонімів: для кожного аналізованого слова сформовано еталонну та кандидатську множини, визначено нижній і верхній пороги косинусної подібності, на яких обчислено точність і повноту відповідно. Така методика враховує реальну поведінку

системи — вибір найближчих векторів серед кількох представлень одного слова — і дає змогу кількісно зіставляти результати на різних етапах розроблення.

За результатами тестування встановлено, що початкова версія системи досягла точності 33,12 % та повноти 21,13 %. Після першого вдосконалення — формування синтаксично-орієнтованого контексту та квантизації моделі до float16 — точність зросла до 40,24 %, а повнота — до 24,93 %. Друге вдосконалення (переранжування кандидатів через порівняння ембедингів речень) мало кращий ефект: точність досягла 68,28 %, а повнота — 49,18 %, що перевищує початкові значення у 2,33 та 2,06 раза відповідно. Нижній поріг косинусної подібності зріс на 26,22 відсоткових пунктів, що свідчить про підвищення якості ранжування кандидатів.

Сформована збірка налічує 1 589 593 записи та 94 604 унікальні леми. Продемонстровано роботу системи через інтерфейс вебзастосунку.

ВИСНОВКИ

Кваліфікаційну роботу присвячено реалізації NLP-системи для автоматичного добору контекстуальних синонімів в українськомовних текстах із поверненням результатів у граматично узгодженій формі.

У першому розділі оглянуто наявні рішення та визначено інструментальну основу для розроблення системи. Встановлено, що BERT у режимі маскованого мовного моделювання не є оптимальним для добору контекстуальних синонімів, а інструмент LanguageTool, попри наявність функції підбору синонімів, не працює з українською мовою в цьому модулі. Показано, що використання моделі paraphrase-multilingual-mpnet-base-v2 є доцільним для розв’язання поставленого завдання, зокрема з огляду на результати дотренування за методом [4], що дало змогу підвищити точність WSD з 73,5 % до 77,9 %.

У другому розділі реалізовано всі компоненти системи й описано закладені в них механізми. Уперше запропоновано метод автоматичного добору контекстуальних синонімів для української мови, що ґрунтується на використанні сформованої з трьох текстових корпусів збірки контекстуальних ембедингів лексем, і розширений видобуванням гіперонімів, що спирається на морфосинтаксичне аналізування дефініцій українськомовного тлумачного словника. Цей метод поєднує багатокроковий конвеєр очищення даних (орфографічна перевірка лем, відсіювання за словником ВЕСУМ, дедуплікація близьких векторів) із застосуванням синтаксично орієнтованого контексту та подальшим переранжуванням на рівні речень для покращення результатів роботи системи. Для кожної з чотирьох підтримуваних частин мови (дієслова, іменника, прикметника та прислівника) реалізовано окремий механізм морфологічного узгодження.

У третьому розділі реалізовано спосіб оцінювання на основі словника синонімів і проведено три раунди тестування. Порівняння результатів засвідчило, що запропоновані вдосконалення (синтаксично орієнтований

контекст і переранжування на рівні речень) сукупно підвищили точність системи у 2,33 раза — до 68,28 % — і повноту у 2,06 раза — до 49,18 %. Аналіз сформованої збірки та візуалізація семантичного простору унаочнили її структуру та зміст.

Отже, розроблена система уможливлює автоматичний добір контекстуальних синонімів для слова в реченні та повертає результати в граматично узгодженій формі, придатній для безпосереднього використання в українськомовному тексті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. draft. 2024. URL: https://web.stanford.edu/~jurafsky/slp3/ (date of access: 01.05.2026).
2. Кочерган М. П. Вступ до мовознавства : підручник. Київ : Акад., 2002. 368 с.
3. Білоусова А. В., Дячук Н. В. Поняття та особливості контекстуальної та лексичної синонімії. *The Philological UniverseS*. 2023. С. 24–26.
4. Laba Y. et al. Contextual Embeddings for Ukrainian: A Large Language Model Approach to Word Sense Disambiguation. *Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP)*. 2023. P. 11–19. URL: https://doi.org/10.18653/v1/2023.unlp-1.2 (date of access: 01.05.2026).
5. Devlin J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT*. 2019. P. 4171–4186. URL: https://doi.org/10.18653/v1/N19-1423 (date of access: 01.05.2026).
6. Демська О. Текстовий корпус: поняття і визначення. *Дивослово*. 2011. № 10. С. 35–37.
7. М. Шведова та ін. Генеральний регіонально анотований корпус української мови (ГРАК). *ГРАК*. URL: https://uacorpus.org/ (дата звернення: 01.05.2026).
8. Fischer S. et al. Contemporary News Corpus of Ukrainian (CNC-UA): Compilation, Annotation, Publication. *Proceedings of the Third Ukrainian Natural Language Processing Workshop (UNLP)*. Torino, Italia, 2024. P. 1–7. URL: https://aclanthology.org/2024.unlp-1.1/ (date of access: 01.05.2026).
9. Nivre J. et al. Universal Dependencies v1: A Multilingual Treebank Collection. *Proceedings of LREC*. 2016. P. 1659–1666. URL: https://aclanthology.org/L16-1262/ (date of access: 01.05.2026).
10. Project to generate POS tag dictionary for Ukrainian language. *GitHub*. URL: https://github.com/brown-uk/dict_uk (date of access: 04.05.2026).

11. Старко В. Ф., Рисін А. Великий електронний словник української мови (ВЕСУМ) як засіб NLP для української мови. *Галактика Слова. Галині Макарівні Гнатюк* / Ін-т укр. мови НАН України. Київ : Вид. дім Дмитра Бураго, 2020. С. 135–141.
12. Словник української мови : в 11 т. / АН УРСР, Інститут мовознавства ім. О. О. Потебні. Київ : Наукова думка, 1970–1980.
13. Словник синонімів. *GitHub*. URL: https://github.com/LinguisticAndInformationSystems/mphdict/wiki/Словник-синонімів (дата звернення: 01.05.2026).
14. Смиш О. Р., Швець Д. В. Аналізування українськомовних віршів засобами обробки природної мови. *Наукові записки НаУКМА. Комп'ютерні науки*. 2025. Т. 8. С. 213–224. URL: https://doi.org/10.18523/2617-3808.2025.8.213-224 (дата звернення: 01.05.2026).
15. Straka M., Hajic J., Stěpánek J. UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. 2016. P. 4290–4297. URL: https://aclanthology.org/L16-1680/ (date of access: 01.05.2026).
16. Kotsyba N., Moskalevskyi B., Romanenko M. UD_Ukrainian-IU. *GitHub*. URL: https://github.com/UniversalDependencies/UD_Ukrainian-IU (date of access: 01.05.2026).
17. Qi P. Et al. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. *Proceedings of ACL: System Demonstrations*. 2020. P. 101–108. URL: https://doi.org/10.18653/v1/2020.acl-demos.14 (date of access: 01.05.2026).
18. spaCy. Industrial-strength Natural Language Processing in Python. *spaCy*. URL: https://spacy.io/ (date of access: 01.05.2026).
19. LanguageTool. *LanguageTool*. URL: https://languagetool.org/ (date of access: 02.05.2026).

20. Multilingual Grammar Checker | LanguageTool Features. *LanguageTool*. URL: https://languagetool.org/insights/post/product-languages-supported/ (date of access: 02.05.2026).
21. How to Paraphrase Sentences Perfectly with LanguageTool: Best Paraphrasing Tool. *LanguageTool*. URL: https://languagetool.org/insights/post/product-rephrase-feature/ (date of access: 02.05.2026).
22. paraphrase-multilingual-mpnet-base-v2. *Hugging Face*. URL: https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2 (date of access: 02.05.2026)
23. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of EMNLP-IJCNLP*. 2019. P. 3982–3992. URL: https://doi.org/10.18653/v1/D19-1410 (date of access: 04.05.2026).
24. XLM-RoBERTa. *Hugging Face.* URL: https://huggingface.co/docs/transformers/model_doc/xlm-roberta (date of access: 04.05.2026).
25. Reimers N., Gurevych I. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020. P. 4512–4525. URL: https://doi.org/10.18653/v1/2020.emnlp-main.365 (date of access: 04.05.2026).
26. UberText 2.0. *lang-uk*. URL: https://lang.org.ua/en/ubertext/ (date of access: 04.05.2026).
27. TripletMarginLoss. *PyTorch Documentation*. URL: https://pytorch.org/docs/stable/generated/torch.nn.TripletMarginLoss.html (date of access: 04.05.2026).
28. Johnson J., Douze M., Jégou H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*. 2019. Vol. 7, № 3. P. 535–547. URL: https://doi.org/10.1109/TBDATA.2019.2921572 (date of access: 04.05.2026).

29. Union-Find Algorithm — Python. *Medium*. URL: https://medium.com/@dhleee0123/union-find-algorithm-with-python-and-optimize-d0fd3431cb18 (date of access: 04.05.2026).
30. Пошук по Великому електронному словнику української мови. *ВЕСУМ*. URL: https://vesum.org/ (дата звернення: 04.05.2026).
31. ONNX Runtime. *ONNX Runtime*. URL: https://onnxruntime.ai/ (date of access: 04.05.2026).
32. Google Colab. *Google Colab*. URL: https://colab.research.google.com/ (date of access: 04.05.2026).
33. FastAPI. *FastAPI*. URL: https://fastapi.tiangolo.com/ (date of access: 04.05.2026).
34. React. *React*. URL: https://react.dev/ (date of access: 04.05.2026).
35. Matplotlib – Visualization with Python. *Matplotlib*. URL: https://matplotlib.org/ (date of access: 05.05.2026).
36. Vega-Altair: Declarative Visualization in Python – Vega-Altair 6.1.0dev documentation. *Vega-Altair*. URL: https://altair-viz.github.io/ (date of access: 05.05.2026).
37. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction – umap 0.5.8 documentation. *UMAP*. URL: https://umap-learn.readthedocs.io/en/latest/ (date of access: 05.05.2026).

Додаток А

Приклад тіла запиту та відповіді сервера за маршрутом «/synonyms»

Запит:


```
{
  "sentence": "Студент поступово усвідомлював складність поставленого завдання.",
  "word": "усвідомлював",
  "min_similarity": 0.85,
  "max_nearest": 10
}
```


Відповідь:


```
{
  "word": "усвідомлював",
  "lemma_used": "усвідомлювати",
  "pos": "VERB",
  "feats": { "Tense": "Past", "Number": "Sing", Gender": "Masc", Mood": "Ind" },
  "query_definition": "[осягав] розумом, [сприймав] свідомо, [розумів] значення, сенс",
  "results": [
    { "word": "розумів", "lemma": "розуміти", "similarity": 0.98 },
    { "word": "осмислював", "lemma": "осмислювати", "similarity": 0.95 },
    { "word": "уявляв", "lemma": "уявляти", "similarity": 0.91 },
    { "word": "тямив", "lemma": "тямити", "similarity": 0.91 },
    { "word": "виявляв", "lemma": "виявляти", "similarity": 0.88 },
    { "word": "сприймав", "lemma": "сприймати", "similarity": 0.87 }
  ],
  "total": 6
}
```