МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

Веб-застосунок для токенізації тексту українською мовою

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконала: студентка 4 курсу, групи КН- 41

Бірюкова А.І.

Керівник Снитюк В.Є

Доктор технічних наук, професор

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ - 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____2022 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Бірюковій Анастасії Ігорівні

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

Веб-застосунок для токенізації тексту українською мовою

затверджена протоколом засідання кафедри від « 23 » грудня 2021 р. № 4

2. Термін здачі студентом закінченого проекту (роботи) 29 травня 2022 року

3. Вихідні дані до проекту (роботи)

Розроблений веб-застосунок , що складається з веб-сторінки та програмного модулю

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

4.1. Аналітичний огляд інсуючих рішень та постановка завдання щодо створення токенізатору

4.2. Проектування системи та архітектури веб-застосунку

4.3. Програмна реалізація веб-застосунку для токенізації тексту українскьою мовою

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

5.1 Вступ, тема, мета. (1 - 2 слайди)

5.2 Аналітичний огляд інсуючих рішень та постановка завдання щодо створення токенізатору (3 - 7 слади)

5.3 Проектування системи та архітектури веб-застосунку (8 - 13 слайди)

5.4 Програмна реалізація веб-застосунку для токенізації тексту українскьою мовою (14-18 слайди)

5.5 Висновки (19 слайди)

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / Снитюк В.Є. /

(підпис) (ініціали та прізвище)

Завдання прийняв до виконання _____ / Бірюкова А.І./

(підпис) (ініціали та прізвище)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1.	Обговорення з керівником постановки завдання та змісту пояснювальної записки	15.02.2022 – 01.03.2022	Виконано
2.	Аналіз предметної області, визначення актуальності роботи, аналіз літературних джерел	02.03.2022 – 19.03.2022	Виконано
3.	Проектування системи та архітектури	20.03.2022 – 10.04.2022	Виконано
4.	Програмна реалізація веб-застосунку для токенізації тексту українською мовою	11.04.2022 – 01.05.2022	Виконано
5.	Оформлення документації та підготовка презентації	02.04.2022 – 31.05.2022	Виконано

Студент _____ /Бірюкова А.І. /

(підпис) (ініціали та прізвище)

Керівник випускної кваліфікаційної роботи _____ / Снитюк В.Є. /

(підпис) (ініціали та прізвище)

Анотація

Бірюкова Анастасія Ігорівна виконала випускну кваліфікаційну роботу на тему «Веб-застосунок для токенізації тексту українською мовою» за спеціальністю 122 - «Комп'ютерні науки».

У даній роботі було досліджені відомі методи обробки природної мови, поставлено задачу для розробки токенізатора тексту українською мовою, спроектовану систему, яка буде виконувати токенізацію, розроблено програмне забезпечення. Результатом роботи є працюючий веб-застосунок для токенізації тексту українською мовою, який може використовуватися як підсистема для проведення досліджень у сфері обробки природної мови .

Ключові слова: аналіз тексту, природна обробка мови, токенізація, продукційні правила, обробка виключень.

Summary

The graduation thesis: «Web application for tokenisation for the text in Ukrainian language» has been completed by **Biriukova Anastasiia** speciality 122 - «Computer Science».

In the graduation thesis the known methods of natural language processing were researched, the task of developing the tokenisation of the text in Ukrainian language was set, the system which will carry out the tokenisation is designed, the software is developed. The result of the thesis is a working web application for tokenisation of the text in Ukrainian language, which can be used as a subsystem for research in the field of natural language processing.

Key words: text analysis, natural language processing, tokenisation, production rules, exception processing.

Зміст

Перелік умовних скорочень

NLP (англ. Natural Language Processing) - обробка природної мови

NLTK (англ. Natural Language Toolkit) - бібліотека для роботи з токенізацією

ПК - персональний комп’ютер

ВСТУП

З появою перших ЕОМ (електронно-обчислювальних машин), людство почало розвиватися у галузі комп'ютерної техніки. З огляду на стрімкий розвиток комп'ютерної техніки та збільшення кількості завдань, які можна розв'язати за допомогою комп'ютерних систем, постало питання у прямій та безпосередній взаємодії між людиною та комп'ютером. З огляду на вищеперераховані проблеми, з'явилася область в комп'ютерних науках, відома як NLP (англ. Natural Language Processing). Дана технологія дозволяє вирішити проблеми розпізнавання природної мови та її правильну інтерпретацію певною програмою. На шляху до розуміння тексту машиною постає множина проблем, а саме: смислової неоднозначності використаних слів, синтаксичної неоднозначності, референційної неоднозначності, наявності непотрібної інформації тощо. На шляху до усунення даних проблем є 6 основних етапів при роботі з обробкою тексту природною мовою (дані етапи будуть розглянуті у 1 розділі випускної кваліфікаційної роботи). Одним із таких етапів є токенізація як процесу поділу тексту на його складові. Задача токенізації є важливою, у тому числі, у зв'язку із розвитком мережі Інтернет та її використання українськими користувачами.

Український сегмент мережі Інтернет розвивається та потребує використання даної технології, адже практичне застосування NLP знаходить місце як на рівні звичайних користувачів — створення чат-ботів, голосових помічників, перекладачів, автовідповідачів, налаштування таргетованої реклами, сортування листів тощо — так і на рівні високотехнологічних розробок у сфері економіки, фінансів, штучного інтелекту.

Темою даної випускної кваліфікаційної роботи є розробка веб-застосунку для токенізації тексту українською мовою на основі аналізу існуючих рішень у даній сфері та застосуванню навичок, отриманих протягом періоду навчання на спеціальності «Комп'ютерні науки».

Метою даної випускної кваліфікаційної роботи є розробка власного токенізатору як підсистеми для подальшої реалізації у проектах з використанням

технології NLP, з можливістю імплементації у більші проекти для створення продуктів, які задовільняють потреби користувачів, та розвитку українського сегменту в мережі Інтернет.

Об'єктом дослідження є процес поділу тексту на його складові, а саме - слова та речення.

Предметом дослідження є правила обробки виключень та усунення неоднозначності використання знаків пунктуації; алгоритми для токенізації.

Унікальність токенізатору полягає у тому, що текст поділяється на окремі слова (токени) і на речення; вирішення проблеми неоднозначності використання знаків пунктуації та представлений у зручному вигляді для користувача.

У даній роботі досліджено існуючі алгоритми токенізації, проведено дослідницьку роботу з обраної предметної області, поставлено завдання та запропоновано його рішення шляхом створення програмного продукту. Застосовано опановані протягом всього періоду навчання знання та навички.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз поширення та використання української мови в Україні

З огляду на останні події в Україні, впроваджуються дії, спрямовані на поширення використання української мови, проведення політики українізації у різних сферах діяльності людини. Проводиться така політика у наступних аспектах:

- Набуття чинності закону України про функціонування української мови як державної; стаття 27 про використання української мови у сфері користувацьких інтерфейсів комп’ютерних програм та веб-сайтів [1].
- Зростання рівня зацікавленості іноземних осіб до української мови, на що вказує стрімкий рівень збільшення вивчення державної мови (на 577 %) у додатку для вивчення іноземних мов Duolingo [12].
- підвищення рівня національної ідентичності як соціального явища, що теж сприятиме стрімкому розвитку використання української мови.

Вищеперераховані аспекти є підставою для створення запиту суспільства до споживання та передачі інформації українською мовою, у тому числі, і в мережі Інтернет. Компанія «Хостмайстер», яка займається наданням послуг з технічного супровіду та адміністрування домену .ua, у 2021 році провела статистичне дослідження для підрахунку кількості реєстрацій та наявних веб-сайтів на даному домені [10]. Результати наведені у таблиці 1.1

Таблиця 1.1

Домен	Усього, шт.	Приріст*, % к1		Продовження, %	% від загальної кількості	Наявність IP-адреси, %	Наявність IPv6-адреси, %
		01.07.2020	01.06.2021				
UA, усього	559378	+3,89 (+0,66)	0,13	-	100	-	-

UA, 2ld	24557	+5,43 (+5,12)	0,41	95,55	4,39	92	26,05
COM. UA	324557	+5,02 (+2,29)	0,15	81,43	58,09	91	28,27
KIEV. UA	38397	-0,69 (-4,66)	-0,09	81,81	6,89	90	29,53

З даної таблиці можна отримати інформацію про наявність 559378 доменних імен, з яких приблизно 90% наразі є в активному користуванні. Ці дані показують, що існує великий потенціал для впровадження автоматизованої комунікації між користувачем та суб'єктом надання послуг, а саме — створення чатботів, голосових помічників, перекладачів, автовідповідачів, налаштування таргетованої реклами, сортування листів тощо. З використанням технології NLP розв'язання вищеописаних задач є не лише актуальною темою, але і прибутковою. Інестиції у проекти, де задіяна дана технологія буде тільки зростати упродовж найближчих років. Нижче продемонстровано стовпчасту діаграму, у якій показано як у майбутньому збільшиться капіталізація сервісів з використанням NLP [11]. (рис. 1.1)

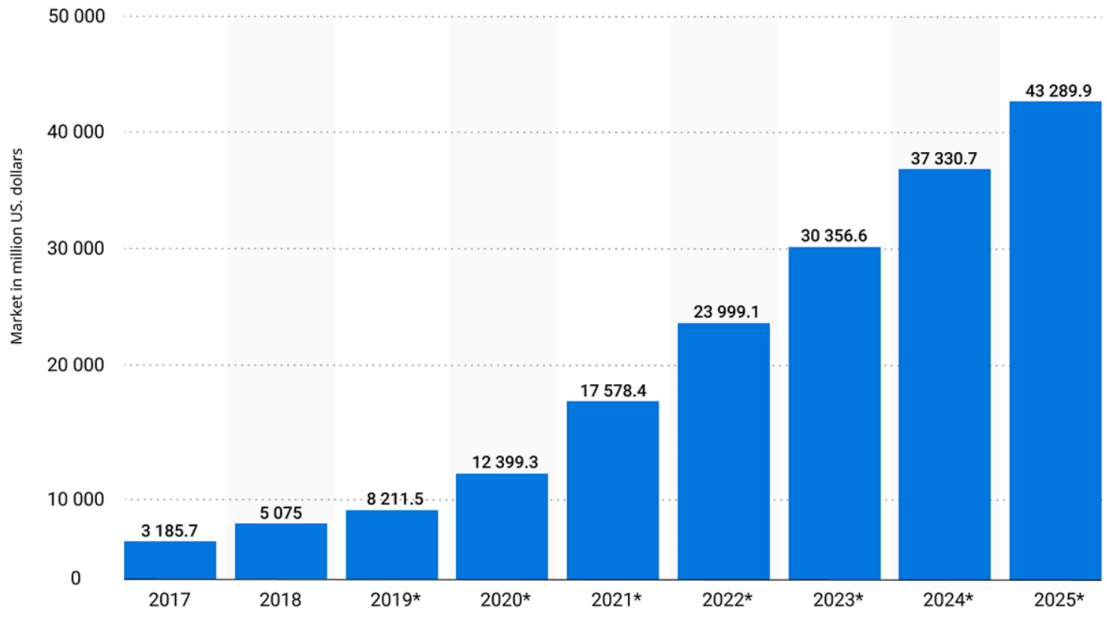


Рисунок 1.1 Діаграма росту капіталізації NLP технологій

1.2 Етапи обробки інформації для технології обробки природної мови

NLP - це область, у якій досліджується обробка природної мови та трансформації, зрозумілої для комп'ютера. Дана галузь знаходиться на перетині лінгвістики, штучного інтелекту та машинного навчання [5].

Можна виділити наступні етапи при роботі з обробкою природньої мови:

1. Сортування документів. Це - процес перетворення набору цифрових файлів у чітко визначені текстові документи. Він може розділятися на декілька етапів,в залежності від файлів, які обробляються. По-перше, для розуміння тексту комп'ютером, необхідно, щоб символи були подані в спеціальному кодуванні. По-друге, необхідно визначити природню мову для подальшого аналізу тексту з використанням конкретних алгоритмів. По-третє, необхідно визначити фактичний зміст документу шляхом видалення непотрібних об'єктів, таких як зображення, таблиці, посилання тощо.

2. Токенізація. Це процес поділу тексту на його складові. Основна задача токенізатора - виділити і розпізнати в письмовому або усному мовленні основні структурні одиниці - лексеми. Поділ тексту буває двох типів: сегментація - поділ тексту на речення, і, власне, токенізація - поділ речень на слова (токени).

3. Стеммінг. Задача даного етапу полягає у відсіканні від слова усіх афіксів - префікса, суфікса, закінчення.

4. Лематизація. Схожий на попередній етап, але є більш уточненим. Його задача полягає у приведенні слова до словникової форми. Для іменника це називний відмінок однини, для дієслова - інфінітив.

5. Нормалізація. Це серія операцій, необхідних для правильного сприйняття тексту комп'ютером, таких як: приведення всіх слів до одного регістру, видалення всіх знаків пунктуації, дешифрування скорочень і т.д.

6. Створення корпусів. Корпус - це вже оброблений набір текстів за певними правилами чи властивостіми. Він є необхідним для проведення лінгвістичного аналізу навчання тощо.

1.3 Опис існуючих рішень

1.3.1 Токенізація та сегментація тексту за допомогою застосування правил та обробки виключень

Даний метод був дуже поширений впродовж 1990-2000 рр. За основу береться принцип, що існує перелік сталих виразів, котрі, при наявності знака пунктуації, створюють єдиний токен або пропис правил, за якими оброблюється певна лексема для створення токена або ж, навпаки, розділення. Крістіна Хоффман, німецька лінгвістка, у 1994 р. використала даний підхід для створення власного корпусу німецької мови. Свій доробок вона виклала в німецькій газеті die tageszeitung. Було створено величезний список абревіатур, які могли класифікувати кінець речення. Протестовано 2827 типів текстів з корпусу і метод показав правильне визначення границь речення у 98% [4].

1.3.2 Сегментація тексту із використанням регресійного дерева

У 1984 році американський вчений Майкл Рілей описав підхід, який використовує за основу принцип регресійного дерева для того, щоб зробити правильну сегментацію. Базується він на наступних ознаках:

- ймовірність того, що слово передує крапці в кінці речення;
- ймоврініть того, що слово слідує після крапки на початку речення;
- довжина слова до крапки;
- довжина слова після крапки;
- випадок слова, що передує крапці (верхній регістр, нижній регістр, всі великі, цифри);
- пунктуація після крапки;
- абревіатура з крапкою.

Метод використовує інформацію про одне слово з можливістю оточення обох боків пунктуаційним знаком, і записати для кожного слова з виключень ймовір-

ність того, що воно зустрінеться разом із кінцем речення. Таким чином було сформовано корпус з 25 млн. слів з точністю 99,8% [4].

1.3.3 Сегментація тексту із використанням нейронних мереж

У якості прикладу наведений алгоритм Satz [4]. Його підхід полягає в тому, щоб показати розділові знаки у кожному реченні як ряд векторів ймовірностей. Дані ймовірності, що використовуються для кожного слова в контексті, формуються із попередньо вирахуваних ймовірностей частин мови, які є отримані з масиву з дескрипторами, що містить дані про частоту використання різних частин мови. Нижче зображено блок-схему роботи алгоритму. (рис. 1.2)

Рисунок 1.2 Зображення блок-схеми алгоритму

1. Спочатку відбувається поділ на токени-слова, які зібрані в одному словнику. Якщо їх немає в описі, є ряд правил, які регулюють дану ситуацію.
2. Відбувається класифікація ймовірності належності токена до конкретної частини мови (іменник, дієслово, прикметник тощо)
3. Створення масиву з дескрипторами. Оскільки існує близько 70-80 частин мови, виділяється основні 18, які і зберігаються в масиві. (рис. 1.3)
4. Сегментація із використанням нейронних мереж. І на виході вже є текст з поділеними реченнями.

noun	verb
article	modifier
conjunction	pronoun
preposition	proper noun
number	comma or semicolon
left parentheses	right parentheses
non-punctuation character	possessive
colon or dash	abbreviation
sentence-ending punctuation	others

Рисунок 1.3 Перелік основних дескрипторів

1.4 Розгляд проблеми через неоднозначність використання знаків пунктуації

Як вже було описано, токенізація - це процес поділу тексту на його складові: слова (токенізатор) чи речення (сегментатор). Оскільки українська мова належить до слов'янської групи мов і має за основу алфавіту кирилицю, тому наявність пробілу свідчить про кінець одного токену і початок другого. Але може виникнути проблема правильного поділу на токени через неоднозначність використання знаків пунктуації. Тобто один знак може виконувати декілька

функцій, що може спричиняти проблеми поділу. Розглянемо проблему сегментації. Наприклад, розділовий знак «крапка» може слугувати для позначення:

а) кінця речення;

б) скорочення;

в) дати;

г) ініціалів;

г) десяткової крапки.

Також може виникнути проблема з використанням знаку оклику. При вигуках він не буде виконувати функцію розділового знаку, а буде частиною токену.

Неоднозначність може виникати при використанні дефісу. Іноді він може означати перенесення слова, іноді — зв’язок складних слів, таких як: ледве- ледве, фізико-математичний, думає-гадає тощо. А іноді - скорочення слів (ін-т - інститут).

Розроблений токенізатор має ураховувати всі особливості використання знаків пунктуації для того, щоб уникнути невідповідностей при поділі тексту.

1.5 Постановка задачі токенізації тексту

1.5.1 Чітке формулювання задачі випускної кваліфікаційної роботи

Темою даної випускної кваліфікаційної роботи є розробка веб-застосунку для токенізації тексту українською мовою на основі аналізу існуючих рішень у даній сфері та застосуванню навичок, отриманих протягом періоду навчання на спеціальності «Комп’ютерні науки».

Метою даної випускної кваліфікаційної роботи є розробка власного токенізатору як підсистеми для подальшої реалізації у проектах з використанням технології NLP, з можливістю імплементації у більші проекти для створення продуктів, які задовільняють потреби користувачів, та розвитку українського сегменту в мережі Інтернет.

Об'єктом дослідження є процес поділу тексту на його складові, а саме - слова та речення.

Предметом дослідження є правила обробки виключень та усунення неоднознозначності використання знаків пунктуації; алгоритми для токенізації.

Майбутній токенізатор базується: на підході створення власних продукційних правил для обробки виключень та усунення неоднозначності використання знаків пунктуації; на використанні вже існуючих правил. Також передбачене використання бібліотеки NLTK (англ. Natural language toolkit) у ході написання алгоритму для процесу токенізації. Дана бібліотека містить у собі набір функцій, використовуючи які, текст розбиається на токени у тих випадках, коли не виникає питання неоднозназності використання певних розділових знаків.

Основною задачею є створення коректно працюючого веб-застосунку для токенізації. Для забезпечення коректної роботи, необхідно визначитися з кроками, які необхідно реалізувати під час розробки:

1. Створити скрипт для реалізації процесу поділу тексту на складові, а саме:
   1.1 скрипт із використання функцій бібліотеки NLTK.
   1.2 скрипт, в основі якого лежать продукційні правила для усунення неоднзначності використання знаків пунктуації.
2. Сформувати датасет, у якому будуть знаходитись скорочення згідно з правилами правопису української мови.
3. Розробити серверну частину, що має реалізувати обробку запити клінта та надсилання відповідей.
4. Розробити клієнтську частину, що має надсилати запити та надавати дані для обробки.
5. Розробити веб-сторінку.

Розроблений застосунок має працювати наступним чином: на вхід подається текст. Існує можливість як завантаження тексту у форматі .docx або .txt, так

і безпосереднього вводу у поле введення.Користувач обирає, який процес необхідно здійснити: токенізації та/або сегментації.

На виході користувач отримує текст, поділений на токени в залежності від обраної функції. Передбачена можливість завантаження поділеного тексту у форматі .txt

Нижче показано дерево задач (рис.1.4), за допомогою якого можна розділити задачі на підзадачі та подати у вигляді ієрархічної структури.

Структура дерева задач є наступною:

1. Розробка бекенду:
   1.1 Розробка алгоритму.
      1.1.1 Написання продукційних правил для обробки виключень.
      1.1.2 Виконання токенізації та сегментації з використанням бібліотеки NLTK.
   1.2 Обробка запитів
2. Розробка фронтенду
   2.1 Розробка дизайну веб-сторінки
   2.2 Розробка інтерфейсу
      2.2.1 Створення запитів до сервера.
      2.2.2 Отримання відповіді на запит

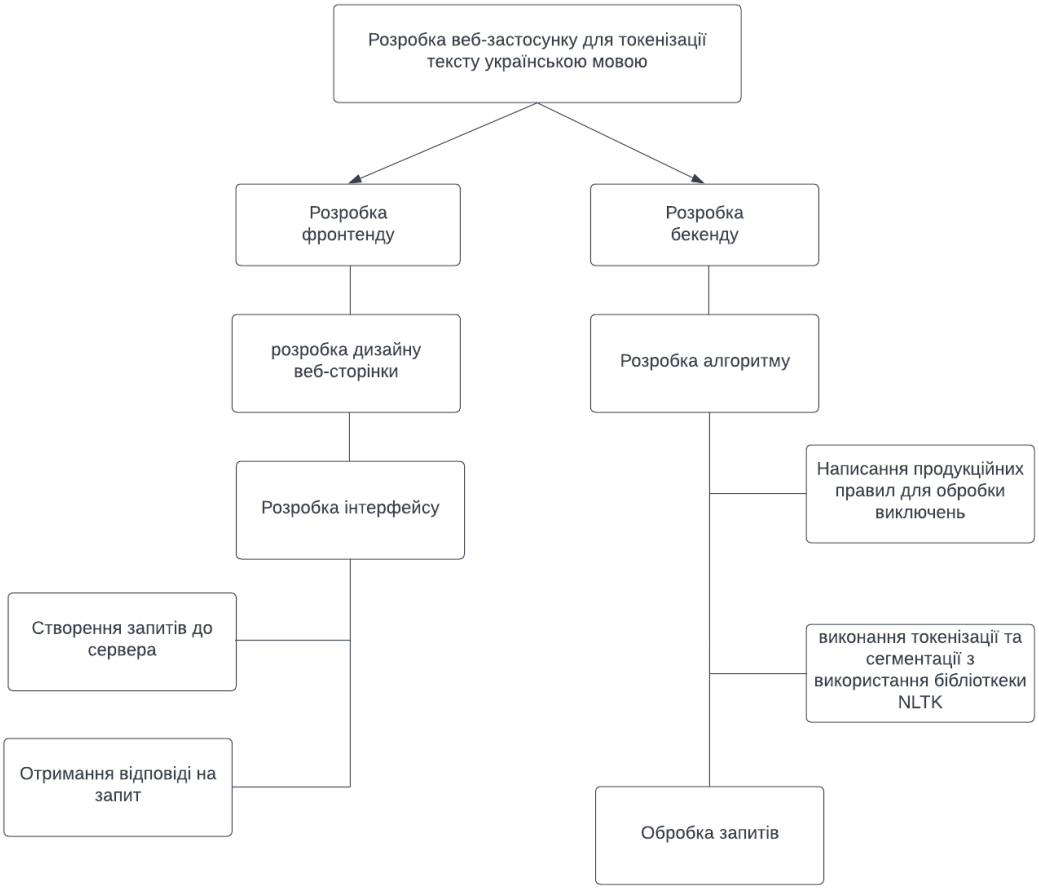


Рисунок 1.4 Дерево задач

1.5.2 Системні вимоги

Системні вимоги – набір характеристик ПК (персонального комп'ютера), які необхідні для ефективної взаємодії користувача та програмного забезпечення. Ілюстрація таких вимог показано у табл. 1.1

Таблиця 1.1 Набір вимог до ПК

Характеристики ПК	Вимоги
Операційна система	Windows 7 і вище, Mac OS X 10.6 і вище

Характеристики ПК	Вимоги
Процесор	Intel Pentium
Оперативна пам’ять	2 ГБ
Місце на диску	200 МБ

1.5.3 Функціональні вимоги

Функціональні вимоги визначають функціональність програмного забезпечення, тобто описують, які можливості має надавати система, що розробляється.

До функціональних вимог можна віднести:

- система має дозволяти збрігати результат на ПК
- клієнт повинен відправляти запит га сервер та отримувати відподь на запит;
- сервер повинен обробляти запити і повертати відповіді на них.

1.5.4 Нефункціональні вимоги

Нефункціональні вимоги описують цілі і атрибути якості. Атрибути якості представляють собою додатковий опис функцій продукту, виражені через опис його характеристик, важливих для користувачів або розробників.

До нефункціональних вимог можна віднести:

- наявність зрозумільго користувацьокого інтерфейсу;
- швидкість обробки запиту не повинна перевищувати 7 секунд.

1.6 Висновок до першого розділу

У першому розділі даної випускної кваліфікаційної випускної роботи було проведено дослідження щодо поширення та використання української мови в мережі Інтернет для визначення актуальності обраної теми випускної кваліфікаційної роботи; було наведено теоретичні відомості про технологію NLP; було досліджено існуючі рішення, а саме три алгоритми для токенізації тексту; розглянуто проблеми, які можуть виникнути під час процесу токенізації через неоднозначність використання знаків пунктуації; було сформовано задачу випускної роботи (тема, мета, об’єкт та предмет дослідження, системні вимоги, функціональні та нефункціональні вимоги), необхідну для подальшого проектування та розробки веб-застосунку.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ТА АРХІТЕКТУРИ ВЕБ-ЗАСТОСУНКУ

2.1 Розробка алгоритму для токенізації тексту

Алгоритми, описані у першому розділі, були створені для токенізації тексту мовами, які належать до германської групи. Українська мова належить до слов'янської групи мов, тому існують деякі відмінності в обробці знаків пунктуації для усунення неоднозначності використання. Для майбутнього токенізатору розроблюється алгоритм, в основі якого лежить застосування правил та обробка виключень, які представлені у вигляді продукційних правил.

Продукційні правила - правила виду «якщо А то В», де А - певна умова, В - дія. [2] Принцип виконання продукційних правил описується наступним чином:

1. Існує умова (факт) А, і він є істинним.
2. Є (правило) Q, яке побудовано наступним чином: якщо А, то В.
3. За умови виконання пунктів 1 і 2, В також стає істинним.

Для розробки алгоритму необхідно написати певну множину правил, які будуть утворювати базу продукційних правил.

Як було описано в першому розділі найчастішою причиною виникнення неоднозначності у використанні знаків пунктуації є використання крапки, особливо для позначення скорочень. Для їх обробки було створено файл, у якому зберігаються скорочення, взяті з академічного словника скорочень української мови []. Слова у скороченому вигляді утворюють певний масив, і вже далі будуть використовуватись в алгоритмі для визначення токенів. Також у даному масиві будуть зберігатись вигуки, тому що вони прописані разом зі знаком оклику, а отже знак пунктуації і слово є одним токеном.

Продукційні правила, вигляду «якщо (умова), то (дія)»:

1. Якщо перед знаком пунктуації та після нього відсутній пробіл, даний знак належить до одного токену зі словом.

2. Якщо розглянутий елемент зі знаком пунткуації належить до словника скорочень, то він є одним токеном.
3. Якщо розглядається знак пунткуації «трикрапка» (…), то даний знак є одним токеном.
4. Якщо перед знаком пунктуації «крапка» (.) та після даного знаку йде літера з великим регістром, то даний знак з літерами є одним токеном.
5. Якщо після знаку пунткуації йде пробіл, то даний знак є окремим токеном.

Окрім продукційних правил, для розробки алгоритму було використано бібліотеку NLTK (англ. Natural language toolkit), яка дозволяє поділяти текст на токени у стандартних ситуаціях, коли не виникає проблем через неоднозначність використання знаків пунктуації.

NLTK - бібліотека мови Python, яка використовується для розробок у сфері NLP. Була розроблена Стівеном Бердом та Едвардом Лопером. [6] Містить широкий набір функцій, використання яких дозволяє розробити алгоритм токенізації.

На основі вищеперерахованих продукційних правил та з урахуванням використання бібліотке NLTK, було створено блок-схему алгоритму, яка продемонстрована нижче (рис 2.3).

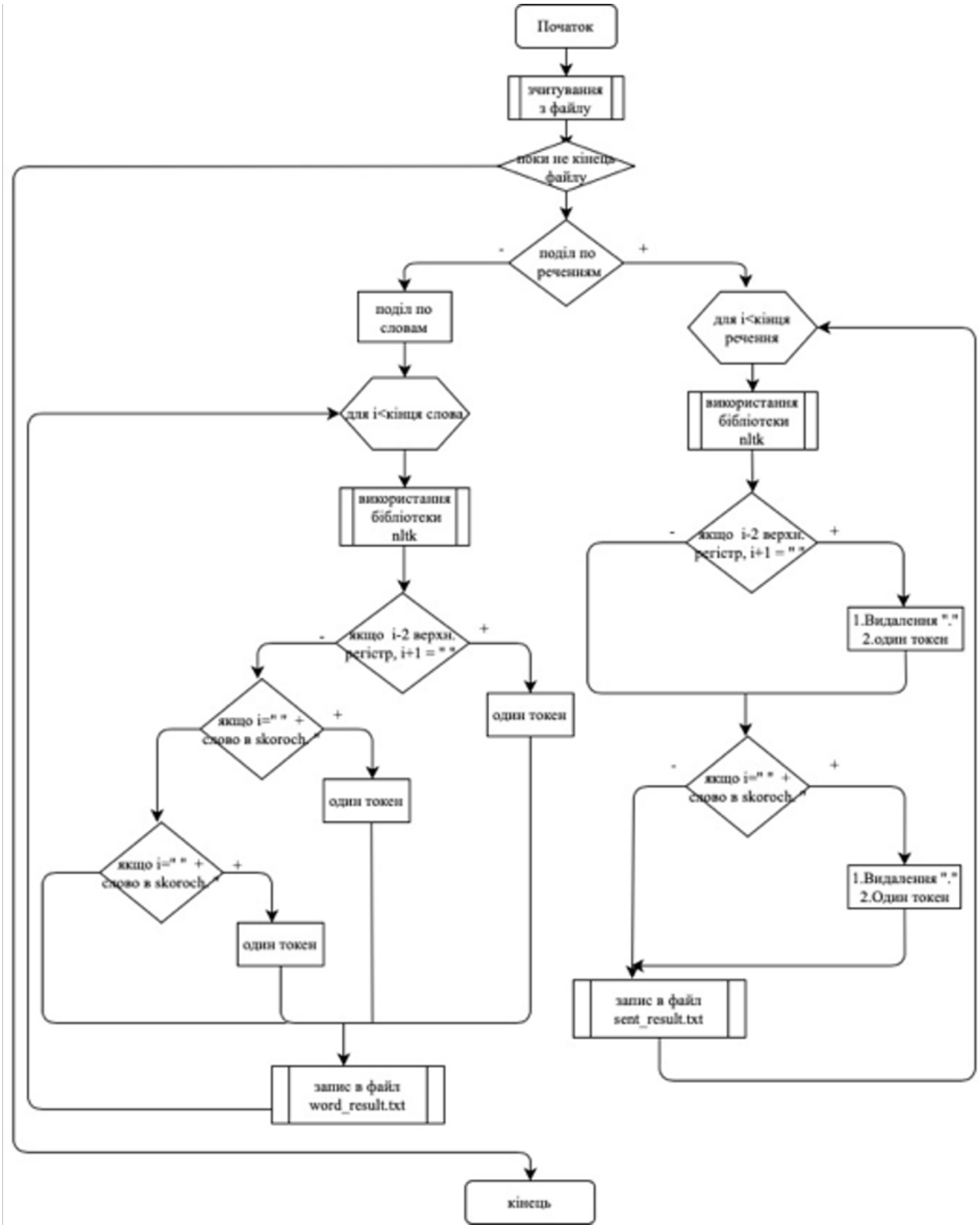


Рисунок 2.1 Блок-схема алгоритму

2.2 Діаграма IDEF0

Наступним етапом у розробці веб-застосунку для токенізації є проектування системи. Для представлення загального опису роботи системи було використано діаграму нотації IDEF0. (рис. 2.2)

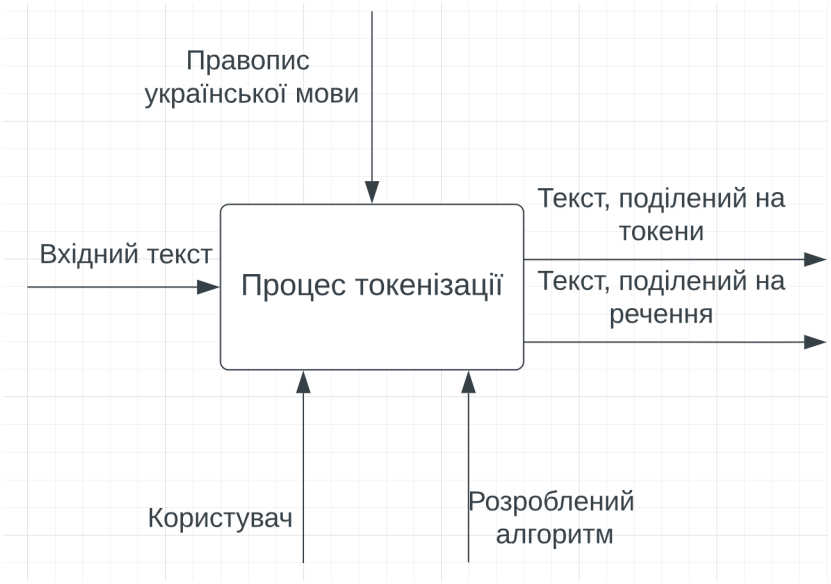


Рисунок 2.2 Діаграма у нотації IDEF0

Нижче подано опис стрілок (ICOM) системи:

1. Вхід (англ. Input) - дані, необхідні для виконання процесу токенізації, - текст, який подається на вхід для подальшої обробки.

2. Керування (англ. Control) - правила чи документи, які регулюють виконання функції - правопис української мови, який регулює використання знаків пунктуації.

3. Механізм (англ. Mechanism) - це особа, пристрій або дані, які виконують функцію - користувач, який подає на вхід текст; розроблений алгоритм, за допомогою якого і відбувається процес токенізації.

4. Виходи (англ. Output) -вихідні дані, отримані в результаті виконання функції, - текст, поділений на токени, та текст, поділений на речення.

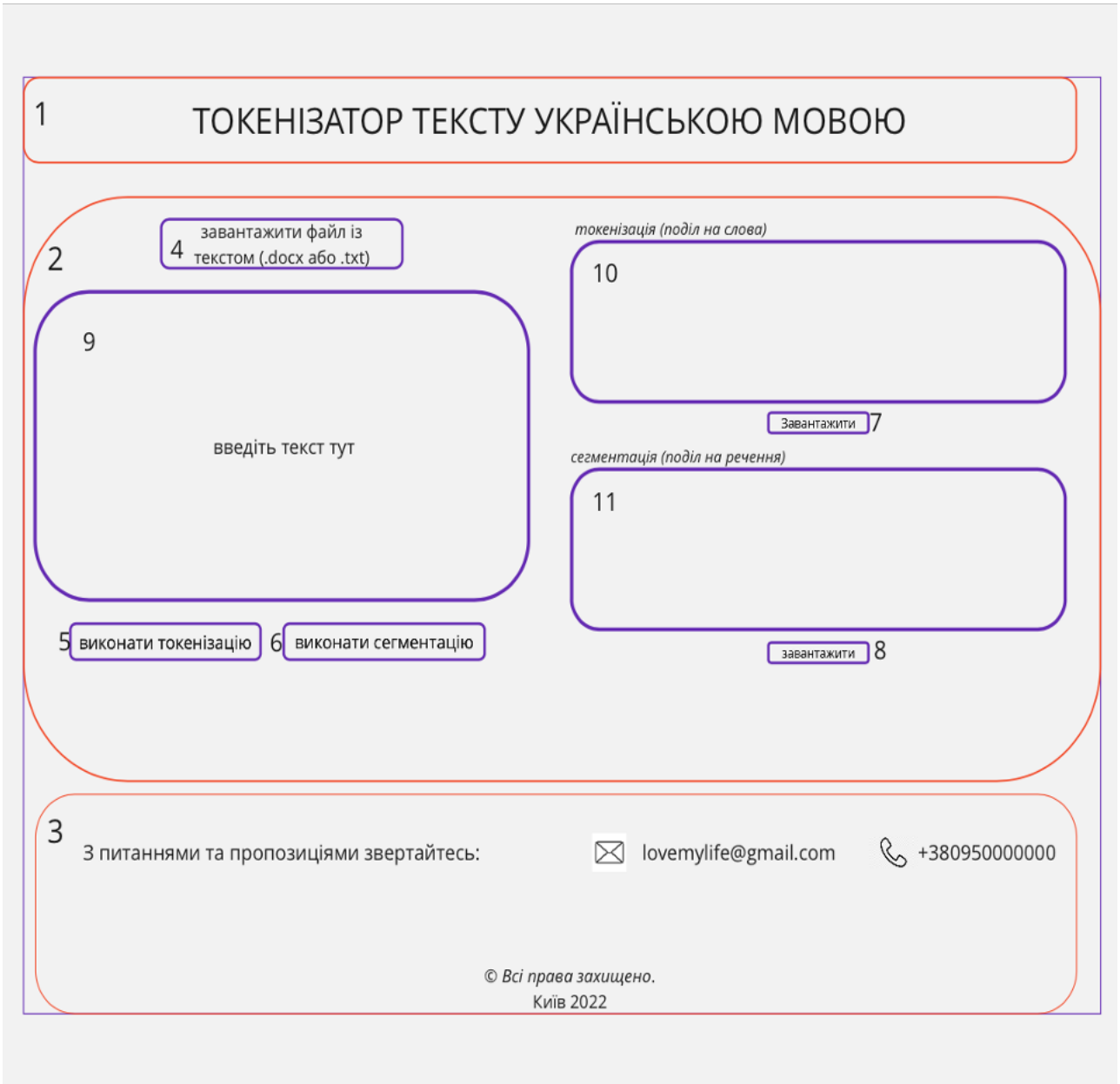
2.3 Узагальнена архітектура системи

Система працює наступним чином: користувач відвідує веб-сторінку, завантажує файл із текстом, який необхідно токенізувати. Після цього до сервера надсилається запит з текстом. Для реалізації алгоритму токенізації, програмний модуль бере дані у форматі json з серверу, відбувається обробка. Для поділу тексту на токени, за необхідності беруться дані з масиву скорочень. Після цього дані у форматі jsonвідправляються на сервер. Надсилається відповідь на запит користувача, і останній отримує результат. Для більш детального розуміння роботи веб-застосунку, нижче продемонстровано зображення архітектури системи (рис. 2.3)

Рисунок 2.3 Узагальнена архітектура системи

2.4 Макет веб-сторінки

Наступним етапом у проектуванні системи є розроблення макету веб-сто-



рінки як каркасу для подальшої розробки веб-сторінки. Даний макет зображено нижче. (рис. 2.4) У таблиці 2.1 подано детальний опис кожного елементу даного макету веб-сторінки.

Рисунок 2.4 макет веб-сторінки

Таблиця 2.1 Опис елементів макету веб-сторінки

№ елементу	Тип елементу	Опис
1	Header	Тег, який відображує заголовок веб-сторінки
2	Body	Тег, який відображує основний контент веб-сторінки
3	Footer	Тег, у якому розташована додаткова інформація
4	Кнопка	Кнопка для завантаження тексту
5	Кнопка	Кнопка для виконання токенізації
6	Кнопка	Кнопка для виконання сегментації
7	Кнопка	Кнопка для завантаження файлу з результатом токенізації
8	Кнопка	Кнопка для завантаження файлу з результатом сегментації
9	Поле	Поле для ручного вводу тексту
10	Поле	Поле для виведення результату токенізації
11	Поле	Поле для виведення результату сегментації

2.5 Висновок до другого розділу

У другому розділі випускної кваліфікаційної роботи було спроектовано веб-застосунок для токенізації тексту українською мовою. Були виконані наступні кроки:

- представлено розрогорнутий опис алгоритму, за яким буде відбуватись токенізації. Надано опис використаних технологій, а саме NLTK. Створено блок-схему роботи алгоритму;
- наведено діаграму у нотації IDEF0 для опису роботи системи;
- побудовано схему, яка відображує узагальнену архітектуру систему і надано словесний опис;
- розроблено макет веб-сторінки, надано словесний опис елементів веб-сторінки.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ТОКЕНІЗАЦІЇ ТЕКСТУ УКРАЇНСЬКОЮ МОВОЮ

3.1 Опис використаних технологій під час розробки

Реалізація програмного модуля була написана мовою програмування python.

Під час розробки було використано бібліотеку NLTK, яка дозволяє обробляти природну мову. Були підключені функції «sent_tokenize, word_tokenize».

Було застосовано фреймворк Flask, необхідний для створення веб-застосунків.

Для реалізації користувальницького інтерфейсу, було використано javascript фреймворк Vue.

Для опису зовнішнього вигляду веб-сторінки було використано формальну мову опису стилів CSS та стандартизовану мову розмітки локументів HTML. (табл. 3.1)

Таблиця 3.1 Опис використаних програмних засобів

№ елемен-	Назва	Логотип	Опис
1	Python		Мова програмування
2	NLTK		Бібліотека NLTK (Natural Language Toolkit),
3	Flask		Фреймворк
4	Vue		Фреймворк
5	CSS		Формальна мова опису стилів
6	HTML		Стандартизована мова розмітки документів

3.2 Фізична структура розробленого веб-застосунку

Елементами розробленої системи є:

Папки:

- project - головна папка, у кій знаходиться проект
- flask_nlp - папка, у якій реалізовано алгоритм токенізації, розроблено сервер для обробки запитів.
- vue_nlp - папка, у якій розроблено користувацький інтерфейс та розроблено функції запитів на сервер.

У папці flask_nlp знаходяться наступні елементи:

- app.py - python файл з розроблено бізнес логіку.
- requirements.txt - файл, у якому прописані бібліотеки для встановлення.
- skoroch.txt - файл, у якому зберігаються скорочення зі словника скорочень.
- result_sentence.txt - файл, у якому зберігається результат поділу тексту на речення.
- result_word.txt - файл, у якому зберігається результат поділу тексту на токени.

У папці vue_nlp знаходяться наступні елементи:

- App.vue - файл з розробленим інтерфейсом користувача.
- main.js - файл, який підключає « App.vue».

Фізична структура у середовищі PyCharm продемонстрована нижче (рис.3.1)

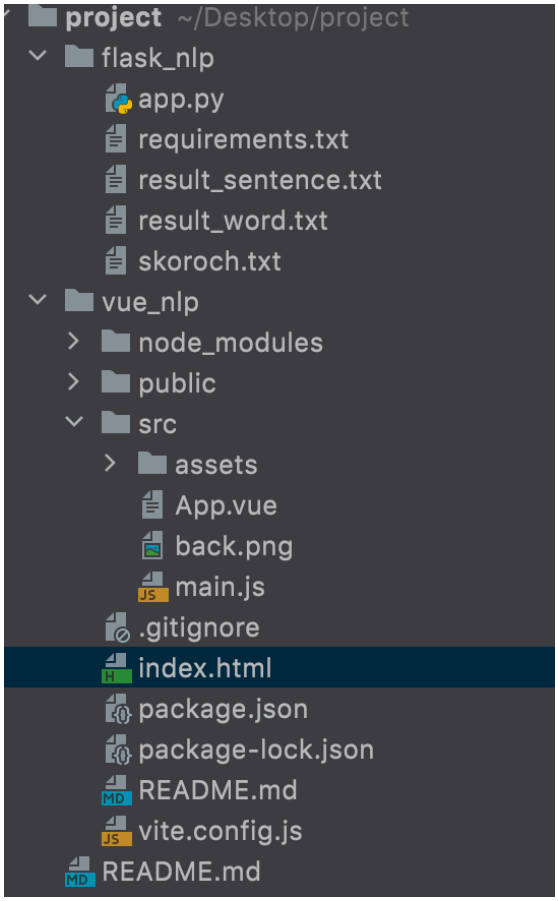


Рисунок 3.1 Фізична структура проекту

3.3 Структура програмного модулю

3.3.1 Структура бекенду

Головним модулем є файл «app.py», у якому реалізована бізнес-логіка. Нижче продемонстровано графічне представлення структури головного програмного модуля. (рис.3.2)

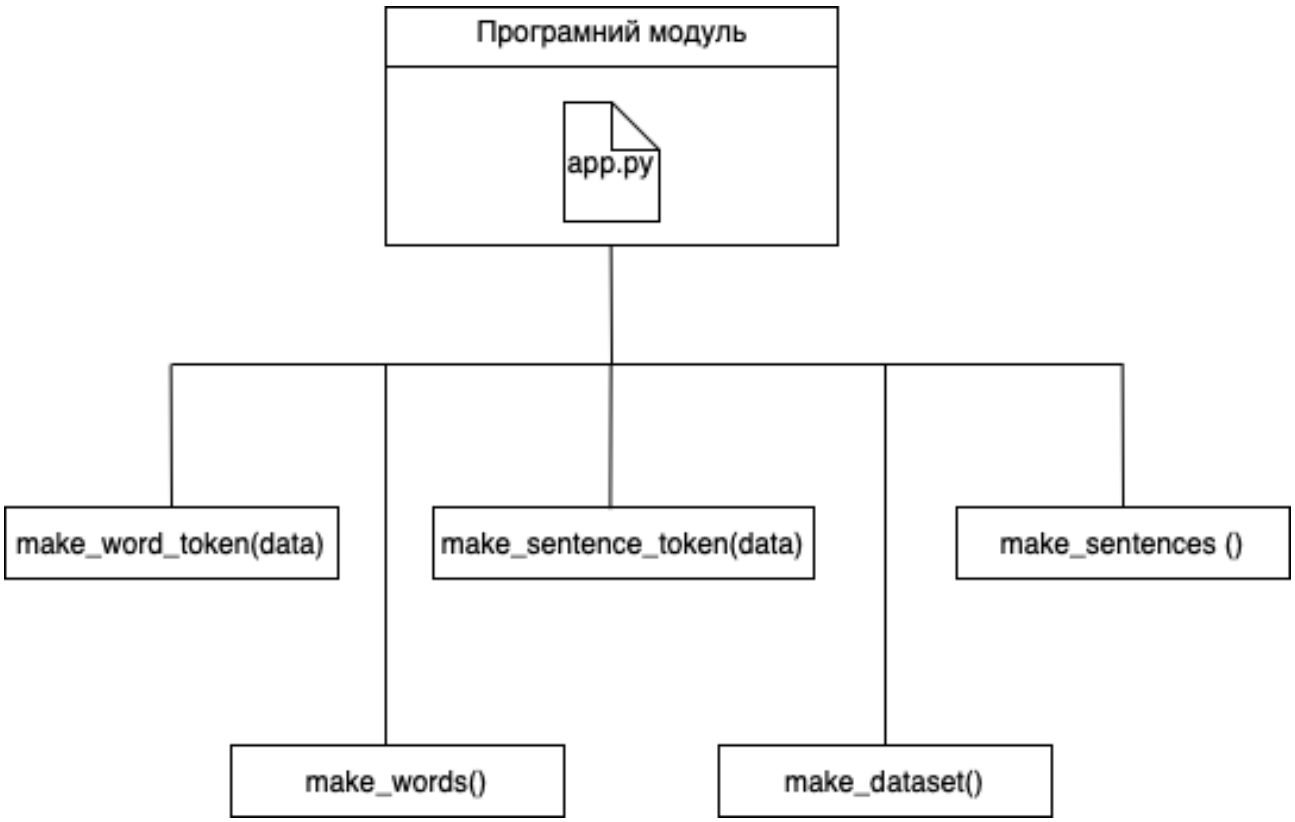


Рисунок 3.2 Графічне предстaлення структури головного програмного модуля

У файлі «app.py» реалізовані 5 наступних основних функцій:

- make_word_token(data) - функція, у якій реалізовано алгоритм поділу тексту на токени.
- make_sentence_token(data) - функція, у якій реалізовано алгоритм поділу тексту на речення.
- make_sentences() - функція, у якій реалізовано процес формування json файлу даними з тексту, поділеним на речення, реалізовано процес передачі даних на сервер.
- make_words() - функція, у якій реалізовано процес формування json файлу даними з тексту, поділеними на слова, реалізовано процес передачі даних на сервер.
- make_dataset() - функція, у якій реалізовано створення датасету із даними з файлу зі скороченнями.

3.2.2 Структура фронтенду

У файлі «App.vue» написано скрипт, який реалізує інтерфейс веб-застовнку за допомогою функцій, а саме:

- function to_tokens(list) - функція, яка перетворює дані у масив.
- async function upload_file(e) - функція, яка вивантажує файл на сервер.
- async funtcion do_words() - функція, яка відправляє запит на токенізацію на сервер.
- async function download_file(name, data) - функція, яка дозволяє завантажити файли на ПК.
- async funtcion do_sentences() - функція, яка відправляє запит на сегментацію на сервер.

Зовнішній вигляд веб-сторінки описано за допомогою CSS та HTML.

Схематичне представлення структури фронтенду продемонстровано нижче. (рис. 3.3)

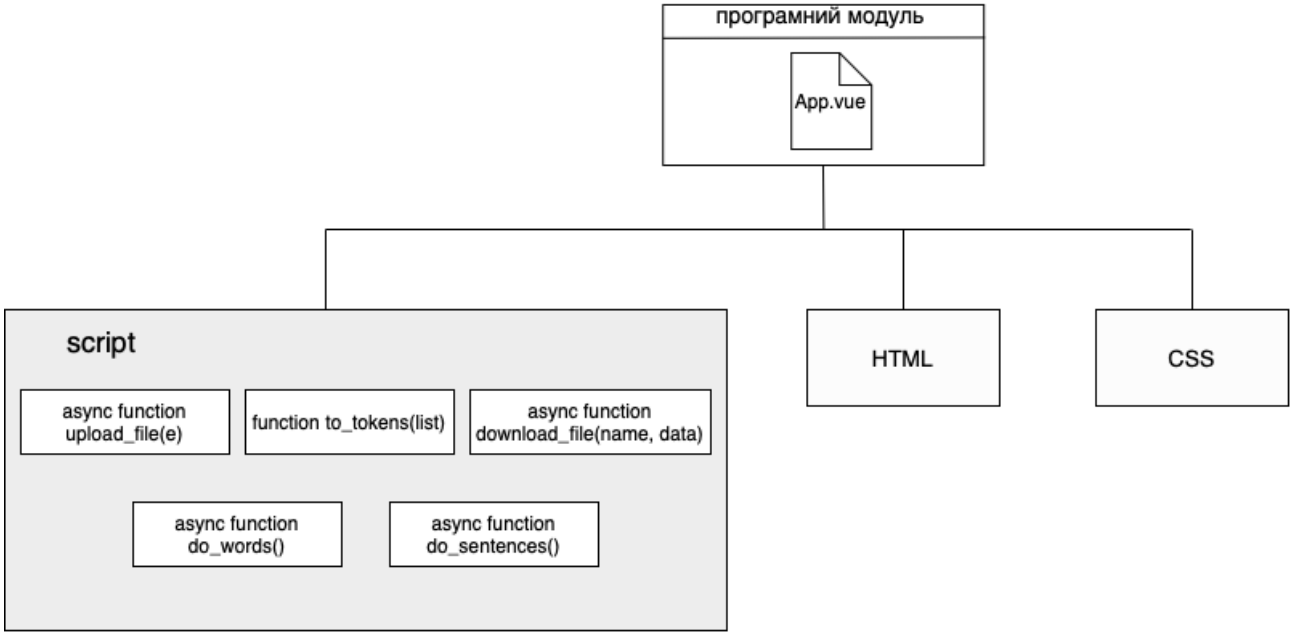


Рисунок 3.3 Схематичне представлення структури фронтенду

3.3 Демонстрація роботи веб-застосунку для токенізації тексту

Щоб запустити веб-застосунок, у терміналі необхідно ввести наступні команди:

У папці flask_nlp: python app.

У папці vue_nlp: nom run dev.

Результат продемонстровано нижче. (рис. 3.4)

Рисунок 3.5 виконання команди на запуск

Після запуску програми, відкривається веб-сторінка. Вигляд веб-сторінки продемонстровано нижче. (Рис.3.5)

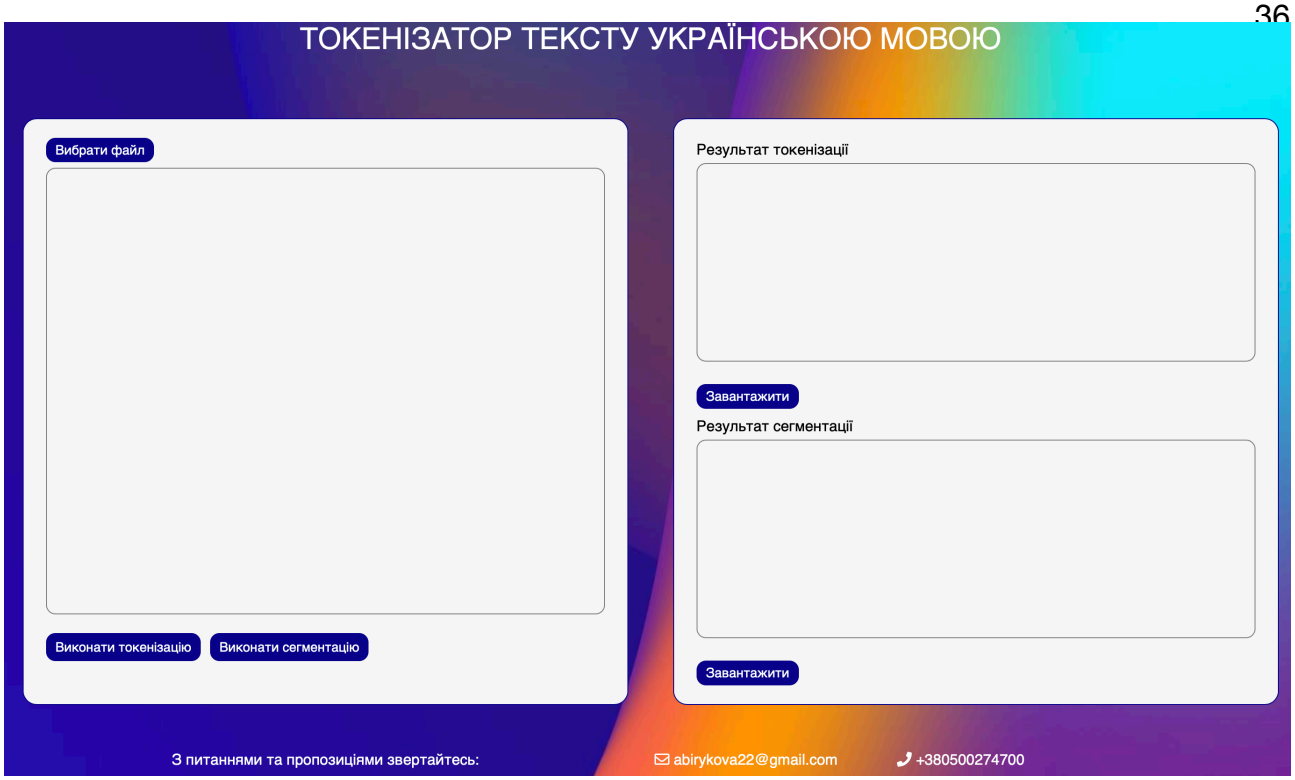


Рисунок 3.5 Вигляд веб-сторінки

Для демонстрації роботи веб-застосунку оберемо текстовий файл, який необхідно буде токенізувати. Завантажимо його у поле вводу за допомогою кнопки «Вибрати файл». (рис. 3.6)

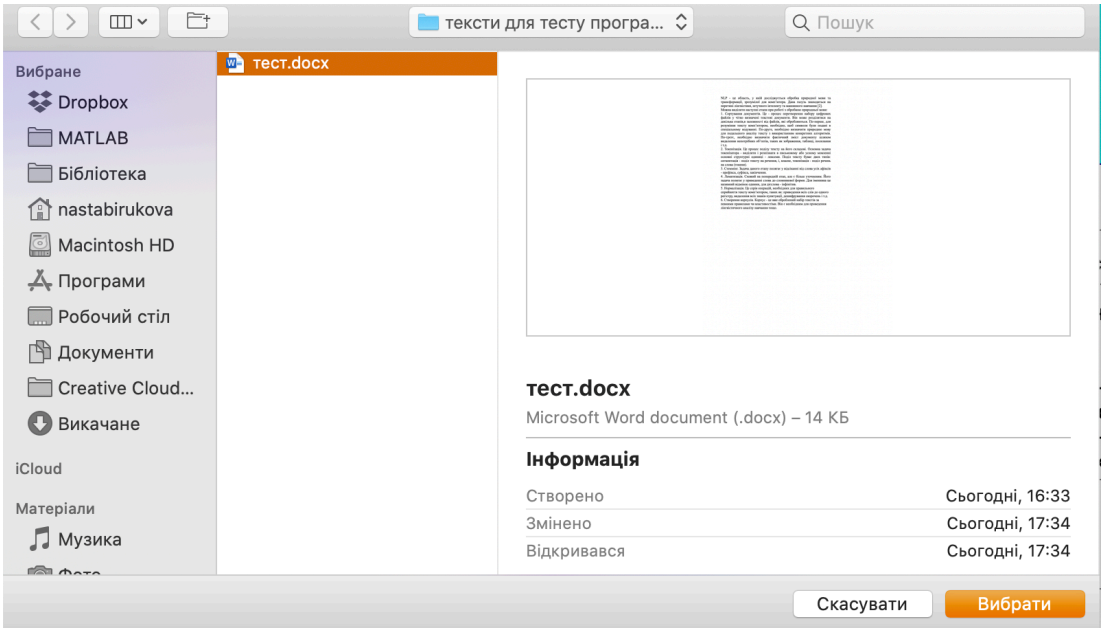


Рисунок 3.6 Завантаження файлу

Після завантаження файлу, його зміст з’являється у полі вводу для зручного перегляду користувачем. При натискані кнопки «Виконати токенізацію», результат з’являється у полі «Результат токенізації». (рис.3.7) При натисканні кнопки «Виконати сегментацію», результат з’являється у полі «Результат сегментації». (3.8)

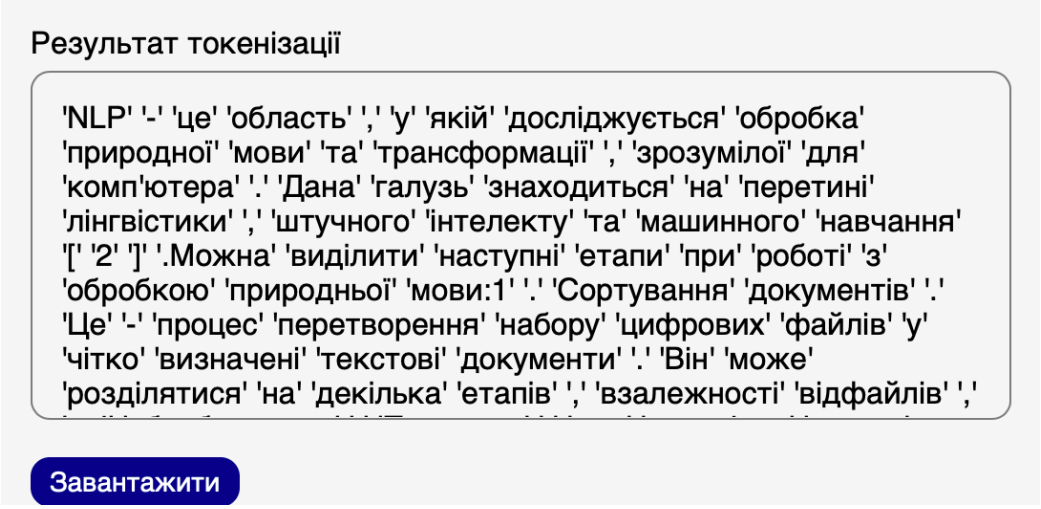


Рисунок 3.7 Результат токенізації

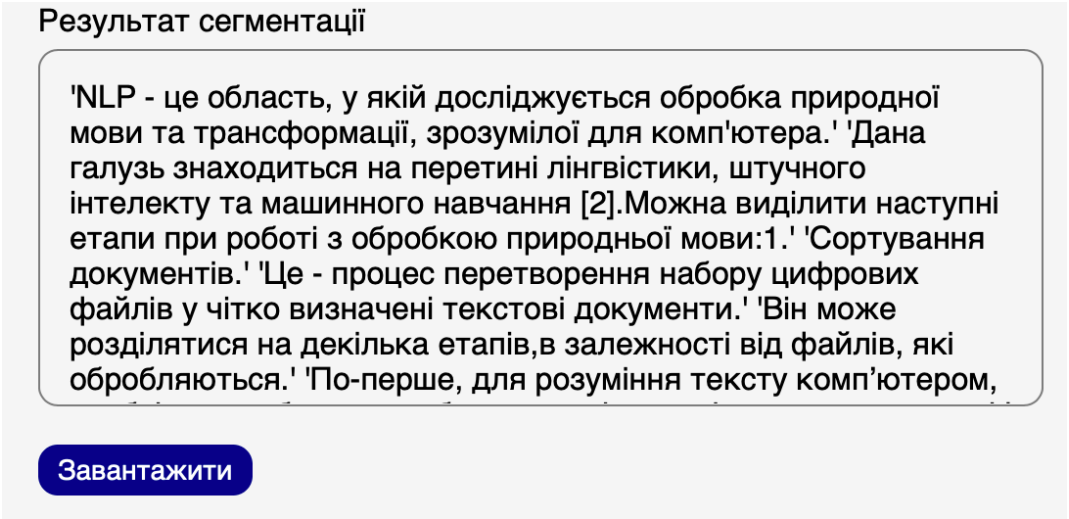


Рисунок 3.8 Результат сегментації

Як видно з рисунку 3.7 та рисунку 3.8, токенізатор працює корректно. Токени відділяються одне від одного за допомогою одирнаних лапок. Для подальшої роботи з текстом, користувач може зберегти результат на власному ПК, натиснувши кнопку «Завантажити». (рис. 3.9)

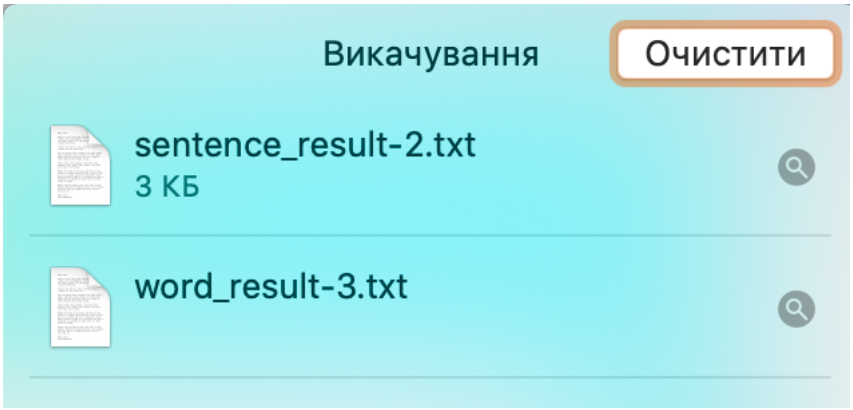


Рисунок 3.9 Завантаження файлів з обробленим текстом

Також перевіримо роботу токенізатора при наявності у тексті конструкцій, які викликають неоднозначність. Для цього у поле вводу надрукуємо текст (рис. 3.10)

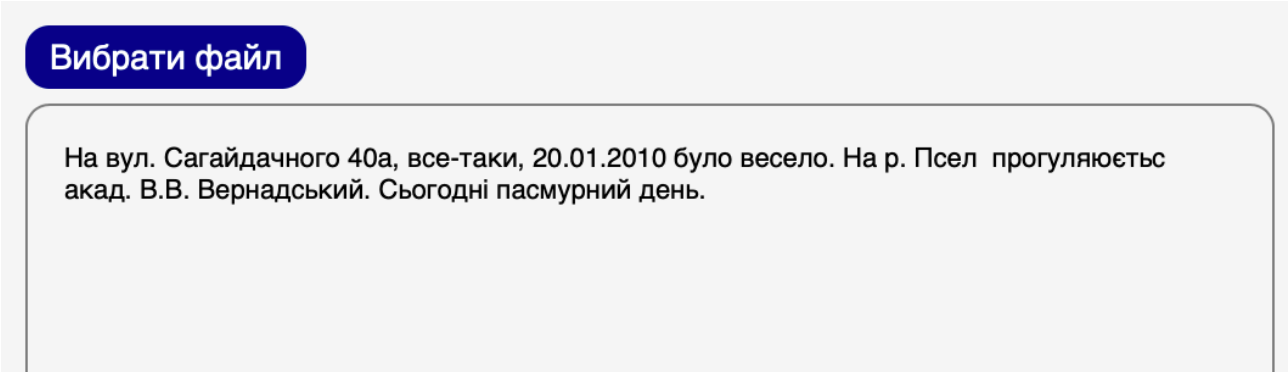


Рисунок 3.10 Надрукований текст у полі вводу

При натисканні кнопки «виконати токенізацію», отримаємо результат, зображений на рис. 3.11. Як видно з результатів, токенізація відбулася успішно. Усі неоднозначності були оброблені, а саме:

1. Скорочення «вул.» - «вул» + «.» є одним токеном.
2. Слово «все-таки», де є використання дефісу є одним токеном.

3. Дата «20.01.2010», яка роздялієтся знаком «.» Є одним токено.
4. Ініціали «В.В» є одним токеном

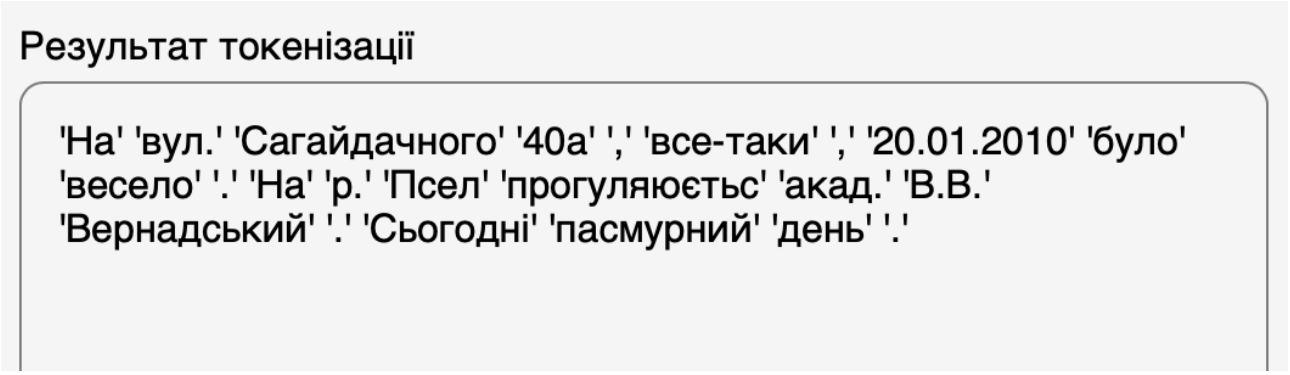


Рисунок 3.11 Результат токенізації

Для перевірки коректності роботи веб-застосунку для токенізації було протестовано 53 тексти різних видів літератури. Процес токенізації та сегментації відбувся корректно.

3.4 Висновок до третього розділу

У даному розділі було продемонстровано програмну реалізацію розробленої системи. Було описано структуру проекту, його складові. Пояснено використання бібліотек та фреймворків при написанні веб-застосунку.

У даному розділі наведено покрокову інструкцію для демонстрації роботи веб-застосунку.

Також було проведено тестування щодо коректності роботи веб-застосунку на вибірці текстів різних типів. Протестовано коректкність роботи застосунку при наявності ситуацій з неоднозначністю використання знаків пунктуації.

ВИСНОВКИ

У результаті виконання випускної кваліфікаційної роботи було розроблено веб-застосунок для токенізації тексту українською мовою.

У першому розділі даної роботи було проведено аналітичних огляд існуючих рішень, а саме три типи алгоритмів для токенізації. Було проведено аналіз щодо поширення використання української мови для визначення актуальності теми роботи; було розглянуто проблеми, які можуть виникнути в результаті неоднозначності використання знаків пунктуації; було поставлено задачу для створення веб-застосунку.

У другому розділі даної роботи було розроблено алгоритм для токенізації тексту; було розроблено систему та прелставлено її узагальнену архітектуру; було розроблено макет веб-сторінки.

У третьому розділі даної роботи було наведено опис програмної реалізації розробленої системи. А саме: було наведено використані технології під час розробки; було показано фізичну структуру проекту і окремих модулів; було продемонстровано роботу веб-застосунку для токенізації тексту українською мовою.

Усі поставлені завдання було виконано. Даний веб-застосунок може знайти своє практичне застосування як підсистема, необхідна для розробки більших проектів у галузі NLP. Даний програмний продукт у майбутньому можна розширювати, функіонал доповнювати розробками інших етапів обробки природної мови. У результаті можна створити великий комерційних проект задля автоматизації багатьох процесів взаємодії людини та комп’ютеру.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://zakon.rada.gov.ua/laws/show/2704-19#Text
2. М.І. Цюцюра, А.В. Єрукаєв, В.В. Гоц, Н.В. Костишина/ Реалізація генетичного алгоритму шляхом застосування продукційних правил// Київський національний університет будівництва і архітектури, Київ, 39 - 2019. 68 с.
3. Л.В. Мухітдінова, Г.М. Пліса, Г.П. Неліпа, О.Б. Зубарєва/ Скорочення слів в українській мові у бібліографічному описі, Київ, 1998. 27 с
4. David D. Planer/ SATZ - An adaptive Sentence Segmentation System, University of California Berkeley 1994. 27 с.
5. S. Bird, E. Klein, E. Loper// Natural language processing with Python// 1005 Gravenstein Highway North, Sebastopol, 2009. 459 с.
6. F. Millstein/ Natural Language Processing With Python: Natural Language Processing Using NLTK, CreateSpace Independent Publishing Platform 2018. 120 с.
7. Є.І. Большакова, Е.С. Клишинський, Д.В. Ланде, А.А. Носков, О.В. Пескова// Автоматическая обработка текстов на естественном языке и компьютерная лингвистика - Москва 2011. 272 с.
8. С. Рассел, П. Норвиг// Искусственный интеллект, современный подход, 2-ое издание - Москва, Киев 2010. 1373 с.
9. М.З. Згуровський, Н.Д. Панкратова// Основи системного аналізу - Київ 2007. 532 с.
10. https://www.hostmaster.ua/news/?stat202106
11. https://www.statista.com/statistics/607891/worldwide-natural-language-processing-market-revenues/
12. https://blog.duolingo.com/duolingo-statement-ukraine/

```
# -*- coding: utf-8 -*-
import os
# Бібліотека для nlp-процессингу
import nltk
# Токени
from nltk.tokenize import sent_tokenize, word_tokenize
import json
# Сервер фреймворк flask
from flask import Flask
from flask import request, jsonify
from flask_cors import CORS

# Бібліотоека для роботи с sql
# from flask_sqlalchemy import SQLAlchemy


app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///
scoroch.sqlite3'
# app.config['']
CORS(app)
# db = SQLAlchemy(app)
nltk.download('punkt')
```

```
punctuation = """!"#$%&''()*+,-./:;<=>?@[\]^_`{|}~—«»"""
```

```
def make_dataset():
    with open("skoroch.txt", "r", encoding="utf-8") as file:
        data = file.read()
        list_scoroch = data.split("\n")
    return list_scoroch
```

```
def read_file():
    file_path = os.path.abspath(entry_file_path.get())
    with open(file_path, "r+", encoding="utf-8") as file:
        data = file.read()
    return data
```

```
def make_word_token(data):
    if not data:
        data = read_file()
    else:
        data = data
    list_index_to_delete = []
    list_words = word_tokenize(data)
    for i in range(len(list_words)):
        for scoroch in scoroch_dataset:
```

```
            if list_words[i]+"." == scoroch:
                list_words[i] = list_words[i]+list_words[i+1]
                list_index_to_delete.append(i+1)
    for i in range(len(list_index_to_delete)):
        del list_words[list_index_to_delete[i]-i]
    list_index_to_delete = []
    for i in range(len(list_words)):
        if len(list_words[i]) >= 2:
            if list_words[i][-1].isupper() and list_words[i]
[-2] == ".":
                list_words[i] = list_words[i] +
list_words[i+1]
                list_index_to_delete.append(i+1)
    for index in list_index_to_delete:
        del list_words[index]
    # print(list_words)
    # with open(file_name, "a+", encoding="utf-8") as file:
    #     for word in list_words:
    #         file.write("'"+word+"' ")
    return list_words


def make_sentence_token(data=False):
    # file_name = "result_sentence.txt"
    if not data:
        data = read_file().replace("«", "`").replace("»",
"`")
```

```
    else:
        data = data
    list_sentences = sent_tokenize(data)
    list_of_elements_to_delete = []
    for i in range(len(list_sentences)):
        if list_sentences[i][-2].isupper():
            list_sentences[i] = list_sentences[i]+" " +
list_sentences[i+1]

list_of_elements_to_delete.append(list_sentences[i+1])
    for index in list_of_elements_to_delete:
        list_sentences.remove(index)
    list_of_elements_to_delete = []
    for i in range(len(list_sentences)):
        for word in scoroch_dataset:
            if " " + word in list_sentences[i]:
                list_sentences[i] = list_sentences[i] + \
                    " " + list_sentences[i+1]

list_of_elements_to_delete.append(list_sentences[i+1])
                break
    for index in list_of_elements_to_delete:
        list_sentences.remove(index)
    list_of_elements_to_delete = []
    # print(list_sentences)
    # with open(file_name, "a+", encoding="utf-8") as file:
    #     for sentence in list_sentences:
```

```
    #         file.write(sentence + "\n\n")
    return list_sentences


def clean_file():
    with open(r"result_word.txt", "w+", encoding="utf-8") as
file:
        file.write("")
    with open(r"result_sentence.txt", "w+", encoding="utf-8")
as file:
        file.write("")


scoroch_dataset = make_dataset()


# POST запрос по адресу http://localhost:5000/api/make_words,
# берет данные для токенизации из данных запроса
@app.post("/api/make_words")
def make_words():
    # Эти данные в формате json которые превратились в пайтон
структуру - словарь
    data = request.json['raw']
    # Из этих данных делаем токены
    word_tokens = make_word_token(data)
    # Формируем обьект с данными: word_tokens - список токе-
нов,
```

```
    # beauty - список токенов соеденненные пробелом в строку
    res = {
        'word_tokens': make_word_token(data),
        'beauty': " ".join(word_tokens)
    }
    # Возвращаем этот обьект на фронт/клиент
    return jsonify(res)
```

```
# POST запрос по адресу http://localhost:5000/api/
make_sentences,
```

```
@app.post("/api/make_sentences")
def make_sentences():
    data = request.json['raw']
    sentence_tokens = make_sentence_token(data)
    res = {
        'sentence_tokens': make_sentence_token(data),
        'beauty': " ".join(sentence_tokens)
    }
    return jsonify(res)
```

```
if __name__ == '__main__':
    app.run(debug=False)
```

Додаток Б

```
<!-- Логика приложения в теге script, "setup" - просто ключе-
вое слово -->
<script setup>
// axios -  бибилитека для http запросов, используется в
do_words(), do_sentences()
import axios from "axios";
// ref - оболочка для переменных
import { ref } from "vue";

// Docxtemplater, PizZip - библиотеки для работы с .docx
import Docxtemplater from "docxtemplater";
import PizZip from "pizzip";

// Переменная с названием хоста сервера
const serverHost = "localhost";
// Переменная с названием порта сервера
const serverPort = "5000";

// Переменная для хранения загруженного на фронт файла
const file = ref(null);
```

```
// Хранение результата ввода
const rawDataInput = ref("");

// Хранение результата токенизации
const word_result = ref("");

// Хранение результата сегментации
const sentence_result = ref("");
// Функция для обработки массивов -
// Слова - ["word1", "word2"] => "word1" "word2"
// Предложения - ["word1", "word2"] => "word1 word2"
// используется в do_words, do_sentences
function to_tokens(list) {
    let res = "";
    for (let i = 0; i < list.length; i++) {
        const element = list[i];
        if (i == list.length - 1) {
            res += `'${element}'`;
        } else {
            res += `'${element}' `;
        }
    }
    return res;
    // return list
}
```

```
// Функция загрузки файла на фронт
async function upload_file(e) {
    word_result.value = "";
    sentence_result.value = "";
    const reader = new FileReader();
    const file = e.target.files[0];
    const fileName = file.name;
    const fileExtenstion = fileName.match(/\.[0-9a-z]+$/i);
    console.log(fileExtenstion[0]);
    if (fileExtenstion[0] === ".docx") {
        reader.readAsArrayBuffer(file);
    } else if (fileExtenstion[0] === ".txt") {
        reader.readAsText(file, "UTF-8");
    }
    reader.onload = function (evt) {
        const readerResult = evt.target.result;
        if (fileExtenstion[0] === ".docx" ||
fileExtenstion[0] === ".doc") {
            const zip = new PizZip(readerResult);
            const doc = new Docxtemplater().loadZip(zip);
            const text = doc.getFullText();
            rawDataInput.value = text;
        } else if (fileExtenstion[0] === ".txt") {
            rawDataInput.value = evt.target.result;
        }
    };
```

```
}

// Функция загрузки результатов
async function download_file(name, data) {
    const a = document.createElement("a");
    const blob = new Blob([
        JSON.stringify(data)
            .slice(1)
            .slice(0, data.length - 1),
    ]);
    a.href = URL.createObjectURL(blob);
    a.download = name;
    a.click();
}

// Функция для запроса на сервер для токенизации rawDataInput
и сохранения данных
async function do_words() {
    // Делаем запрос
    const res = await axios.post(
        `http://${serverHost}:${serverPort}/api/make_words`,
        { raw: rawDataInput.value }
    );
    // Записываем результат в переменную
    const words = res.data.word_tokens;
    word_result.value = to_tokens(words);
```

```
    // Очищаем результат сегментации если есть
    sentence_result.value = "";
}

// Функция для запроса на сервер для сегментации rawDataInput
и сохранения данных
async function do_sentences() {
    // Делаем запрос
    const res = await axios.post(
        `http://${serverHost}:${serverPort}/api/
make_sentences`,
        { raw: rawDataInput.value }
    );
    // Записываем результат в переменную
    const sentences = res.data.sentence_tokens.map((element)
=> {
        return element.replace(/[\r\n]/gm, "");
    });
    sentence_result.value = to_tokens(sentences);

    // Очищаем результат токенизации если есть
    word_result.value = "";
}
</script>

<!-- В template тэге записана вся html разметка -->
```

```
<template>
    <div class="page">
        <div class="page__header header">Токенізатор тексту
українською мовою</div>
        <div class="page__content content">
            <div class="column columns__first">
                <div class="upload__header">
                    <div class="upload__button">
                        <label for="file">
                            Вибрати файл
                            <input
                                type="file"
                                id="file"
                                ref="file"
                                v-on:change="upload_file"
                            />
                        </label>
                    </div>
                </div>
                <div class="raw__text">
                    <textarea v-model="rawDataInput"></
textarea>
                </div>
                <div class="text__buttons">
                    <button @click="do_words" class="button">
                        Виконати токенізацію
```

```
                    </button>
                    <button @click="do_sentences"
class="button">
                        Виконати сегментацію
                    </button>
                </div>
            </div>
            <div class="column columns__second">
                <div class="output__header">Результат токені-
зації</div>
                <div class="output">
                    <div
class="output__content">{{ word_result }}</div>
                </div>
                <button
                    @click="download_file('word_result.txt',
word_result)"
                    class="button"
                >
                    Завантажити
                </button>
                <div class="output__header">Результат сегмен-
тації</div>
                <div class="output">
                    <div
class="output__content">{{ sentence_result }}</div>
                </div>
```

```
                <button
                    @click="
                        download_file('sentence_result.txt',
sentence_result)
                    "
                    class="button"
                >
                    Завантажити
                </button>
            </div>
        </div>
        <footer>
            <div class="contacts">
                <div>
                    З питаннями та пропозиціями звертайтесь:
                </div>
                <div class="contacts-info">
                    <div>
                        <i class="far fa-envelope"></i>
abirykova22@gmail.com
                    </div>
                    <div>
                        <i class="fas fa-phone"></i> +380500274700
                    </div>
                </div>
            </div>
```

```
            <div class="rights">
                © Всі права захищені <br>
                2022
            </div>
        </footer>
    </div>
</template>

<!-- В style тэге записаны все стили для разметки -->
<style>
body {
    height: 100%;
}

#app {
    height: 100%;
    background: url(back.png);
    background-size: 100vw 200vh;
    background-repeat: repeat-x;
    background-attachment: fixed;
}

.page {
    display: grid;
    grid-template-rows: 100px 1fr;
    grid-template-columns: 1fr;
```

```
    margin-left: 24px;
    margin-right: 24px;
    padding-bottom: 24px;
}
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: Helvetica, sans-serif;
   /*font-family: Monaco, monospace;*/
}

.content {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(400px,
1fr));
    /*max-height: 120vh;*/
   /*height: 80%;*/
}

.page__header{
   text-transform: uppercase;
   text-align: center;
   font-size: 32px;
   color: white;
   margin-top: 20px;
```

```
}

.columns {
    display: flex;
    flex-direction: row;
}

.column {
    display: flex;
    flex-direction: column;
   min-height: 350px;
    height: 75vh;
    margin: 24px 24px 50px 24px;
    padding: 24px 24px 10px 24px;
   border: 1px solid darkblue;
   /*box-shadow: white 10px 2px 10px;*/
   border-radius: 15px;
   background-color: whitesmoke;
}

.columns__second {
    display: flex;
    flex-direction: column;
}
#file{
   display: none;
```

```
}
.button, label  {
    height: 30px;
    cursor: pointer;
   color: white;
   background-color: darkblue;
   border: none;
   border-radius: 10px;
   font-size: 14px;
   padding: 5px 10px;
   margin-bottom: 10px;
   margin-right: 10px;
   width: fit-content;
}

.raw__text {
   height: 85%;
   min-height: 200px;
   margin-bottom: 5px;
}

textarea {
   margin-top: 10px;
   padding: 15px;
   height: 95%;
   width: 100%;
```

```
    border: 1px solid gray;
    border-radius: 10px;
    background-color: transparent;
    resize: none;
    outline: none;
}

.output {
    max-height: 500px;
     height: 50%;
     overflow-y: auto;

}
.output__content{
    margin-top: 5px;
    margin-bottom: 5px;
    border: 1px solid gray;
    border-radius: 10px;
    background-color: whitesmoke;
    padding: 15px;
    height: 88%;
}

.output__content {
     overflow-y: auto;
}
```

```
footer{
    color: white;
    text-align: center;
}
.contacts{
    display: grid;
    grid-template-columns: 1fr 1fr;
    margin-bottom: 45px;
}
.contacts-info{
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(50px,
200px));
    text-align: right;
}
.rights{
    font-size: 13px;
    font-style: italic;
}
</style>
```