ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка веб-застосунку з аналізом емоційного стану контексту користувача чату або інформаційного каналу використовуючи мову програмування PYTHON»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки

освітньо-професійної програми Штучний інтелект

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Максим ЯКУСЕВИЧ

Виконав: здобувач вищої освіти гр. ШІД-41
Максим ЯКУСЕВИЧ

Керівник: Максим ФЕСЕНКО
к.т.н., доцент,

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Якусевич Максим Вадимович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Розробка веб-застосунку з аналізом емоційного стану контексту користувача чату або інформаційного каналу використовуючи мову програмування PYTHON»

керівник кваліфікаційної роботи Максим ФЕСЕНКО к.т.н., доцент,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» 02.2024р. № 36

2. Строк подання кваліфікаційної роботи «31» травня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, специфікації та документація з HTML, CSS, JavaScript, JSON, Python та SQL, вимоги створення аналізатора користувача в чаті або інформаційному каналі.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Огляд технологій для розробки веб-застосунку з аналізом емоційного стану контексту користувача чату або інформаційного каналу використовуючи мову програмування PYTHON.

Вибір технологій для реалізації аналізатора текстів: HTML, CSS, JavaScript, JSON, Python та SQL.

Архітектура аналізатора текстів: .

Інтеграція з базою даних та аналізатором

Забезпечення безпеки та масштабованості системи

5. Перелік графічного матеріалу: *презентація*
   1. Схема архітектури системи
   2. Інтерфейс користувача онлайн-консультант
   3. Приклад коду клієнтської та серверної частини

6. Дата видачі завдання «27» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури для розробки веб-застосунку з аналізом	27.02-05.11.23	
2	Перегляд всіх технологій для визначення емоційного стану тексту	05.11-12.11.23	
3	Вибір технологій для реалізації онлайн-консультанта: HTML, CSS, JavaScript, JSON, Python та SQL	13.11-19.11.23	
4	Постановка плану для виконання кваліфікаційної роботи	20.11-25.11.23	
5	Дослідження взаємодій з базою даних та користувача	27.11-03.12.23	
6	Розробка аналізатора на веб-застосунку	04.12-10.12.23	
7	Оформлення роботи: вступ, висновки, реферат	11.12-20.12.23	
8	Розробка презентаційних матеріалів для показу	21.12-31.05.24	

Здобувач вищої освіти _____ Максим ЯКУСЕВИЧ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Максим ФЕСЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 69 стор., 20 рис., 15 джерел.

Мета роботи: Надати розроблений веб-застосунку для аналізування емоційного стану контексту користувача чату або інформаційного каналу використовуючи для розуміння емоційного стану спілкування або інформації

Об'єкт дослідження: Підвищити розуміння емоційного стану чату або інформаційного каналу

Предмет дослідження: Веб-застосунок для розуміння емоційного стану чату або інформаційного каналу.

Короткий зміст роботи: У роботі проведено огляд існуючих технологій розробки веб-аналізаторів, зокрема веб-технологій, таких як HTML, CSS, JavaScript, JSON, SQL, Bootstrap, JQuery, Python . Детально описано вибір та обґрунтування використання цих технологій для створення оптимального веб-аналізатора. Розроблено базу даних, що включає зручне та інтуїтивно зрозумілий зберігання запитів користувачів, а також серверну частину, що забезпечує обробку запитів. Проведено тестування функціональності та збереження даних користувачів для зручності. На основі зібраних даних та відгуків користувачів здійснено оцінку ефективності системи в реальних умовах, що підтвердило її здатність підвищувати рівень розуміння один одного в каналі та інформаційного каналу.

КЛЮЧОВІ СЛОВА: АНАЛІЗ, ВЕБ-АНАЛІЗАТОР HTML, CSS, JAVASCRIPT, JSON, SQL, BOOTSTRAP, JQUERY, PYTHON, ВЕБ-ЗАСТОСУНОК, ЛЮДИ.

ЗМІСТ

Вступ

Вступ до теми “Розробка веб-застосунку з аналізом емоційного стану контексту користувача чату або інформаційного каналу” використовуючи мову програмування Python:

За даними DataReportal, типовий користувач проводить у соціальних мережах близько 2,5 годин на день.

Це 864 години на рік прокручування додатків, що еквівалентно 36 дням або трохи більше місяця.

Коли ми продовжуємо цей часовий проміжок протягом життя, цифри стають ще більш приголомшливими. Якщо припустити, що середня тривалість життя становить 72 роки, а вік 16 років є критерієм для використання соціальних мереж, користувачі можуть провести в соціальних мережах близько 5,5 років свого життя.

Причини такої тривалої витрати часу різні. Соціальні медіа — це платформи для спілкування, розваг і отримання найсвіжішої інформації.

Проте дослідження показали, що повсюдне використання соціальних мереж пов’язане з негативним впливом на психічне здоров’я. Одне дослідження виявило, що тривале використання соціальних мереж пов’язане зі збільшенням симптомів депресії у підлітків.

“Будь-який алгоритм соціальних мереж, чи то Instagram, Facebook або Snapchat, працює практично однаково. Вони оптимізовані для взаємодії”, — сказала авторка Кеті О’Ніл Al Jazeera. “Вони оптимізовані, щоб утримувати вас на платформі соціальних мереж, незважаючи ні на що. Instagram чи Facebook — байдуже, чи це якісно, чи шкідливо, чи ще щось” [13].

У сучасному світі, де цифрові технології стають все більш важливими, а розуміння емоційного стану користувачів може відігравати ключову роль у покращенні їх досвіду використання веб-застосунків. Ця робота фокусується на розробці веб-застосунку, який використовує Python для аналізу емоційного стану користувача на основі контексту його повідомлень у чаті або інформаційному каналі.

Python - це високорівнева мова програмування, яка широко використовується для розробки веб-застосунків та аналізу даних. Вона має багато бібліотек, таких як Flask для розробки веб-застосунків, та NLTK та TextBlob для обробки природної мови, що робить її ідеальним вибором для цього проєкту .

Ми розглянемо, як можна використовувати Python для збору, обробки та аналізу даних з чату або інформаційного каналу, а також для виявлення та аналізу емоційного стану користувача. Ми також обговоримо, як результати цього аналізу можуть бути використані для покращення взаємодії з користувачами та підвищення задоволеності користувачів.

Ця робота спрямована на те, щоб допомогти читачам краще зрозуміти, як можна використовувати Python для розробки веб-застосунків, які аналізують емоційний стан користувачів, та як це може вплинути на взаємодію з користувачами.

Після вступу, ми перейдемо до основної частини розробки, яка буде поділена на декілька ключових розділів:

1. Теоретичні основи: У цьому розділі ми розглянемо основні теоретичні концепції, які лежать в основі аналізу емоційного стану. Це включає в себе поняття емоцій, їх класифікацію, а також методи та техніки, які використовуються для їх виявлення та аналізу
2. Технології та інструменти: У цьому розділі ми детально розглянемо технології та інструменти, які використовуються для розробки веб-застосунку та аналізу емоційного стану. Ми розглянемо такі технології, як обробка природної мови (Natural language processing NLP), машинне навчання, алгоритми класифікації тощо.
3. Практична реалізація: У цьому розділі ми розглянемо практичні аспекти розробки веб-застосунку. Ми обговоримо процес проектування, розробки, тестування та впровадження застосунку. Ми також розглянемо приклади використання застосунку та його можливий вплив на користувачів.

4. Висновки та перспективи розвитку: У заключному розділі ми підведемо підсумки нашого дослідження, вкажемо на основні висновки та обговоримо можливі напрямки для подальшого дослідження та розвитк

1 ТЕОРЕТИЧНІ ОСНОВИ

Давайте розглянемо теоретичні основи розробки веб-застосунку з аналізом емоційного стану контексту користувача чату або інформаційного каналу:

Емоційний стан: Емоційний стан - це комплекс почуттів, які відображають відношення до себе, до інших людей, до подій і предметів. Він може бути позитивним, негативним або нейтральним. Емоційний стан впливає на сприйняття, мислення, поведінку та взаємодію людини.

Аналіз тексту: Аналіз тексту - це процес перетворення неструктурованих текстових даних в структуровану форму для подальшого аналізу. Він включає в себе такі методи, як видалення зайвих символів, стемінг, лематизація, видалення стоп-слів тощо.

Обробка природної мови (Natural language processing NLP): Обробка природної мови - це галузь штучного інтелекту, яка зосереджена на взаємодії між комп’ютерами та людською мовою. Вона використовується для аналізу, розуміння та генерації природної мови.

Машинне навчання: Машинне навчання - це метод штучного інтелекту, який використовує статистичні техніки для навчання комп’ютера виконувати завдання без явного програмування. В контексті аналізу емоційного стану, машинне навчання може бути використано для класифікації тексту на основі емоційного контексту.

Python та його бібліотеки: Python - це високорівнева мова програмування, яка широко використовується для розробки веб-застосунків та аналізу даних. Вона має багато бібліотек, таких як Flask для розробки веб-застосунків, та NLTK для обробки природної мови, що робить її ідеальним вибором для цього проекту.

Ці теоретичні основи допоможуть нам краще зрозуміти, як можна розробити веб-застосунок для аналізу емоційного стану користувача. З нетерпінням чекаю на ваші думки та обговорення цієї захоплюючої теми.

1.1 Емоційний стан

Емоційний стан - це внутрішній стан особистості, який характеризується певними почуттями та емоціями. Він відображає наше відношення до себе, до інших людей, до подій і предметів. Емоційний стан може бути позитивним, негативним або нейтральним, в залежності від нашого сприйняття та реакції на різні ситуації.

Позитивний емоційний стан: Коли ми відчуваємо позитивні емоції, такі як радість, щастя, задоволення, любов, надія, гордість, вдячність тощо, ми перебуваємо в позитивному емоційному стані. Це стан, коли ми відчуваємо задоволення від життя, наснагу до дій та оптимізм щодо майбутнього.

Негативний емоційний стан: Негативний емоційний стан відбувається, коли ми відчуваємо негативні емоції, такі як гнів, страх, смуток, розпач, відчай, заздрість, ненависть тощо. Це може впливати на наше сприйняття, мислення та поведінку, часто ведучи до стресу, тривоги та депресії.

Нейтральний емоційний стан: Нейтральний емоційний стан - це стан, коли ми не відчуваємо ні позитивних, ні негативних емоцій. Це може бути стан спокою, рівноваги або байдужості.

Емоційний стан впливає на наше сприйняття, мислення, поведінку та взаємодію з іншими. Він впливає на наші рішення, наші відносини, нашу продуктивність та загальне благополуччя. Розуміння та управління нашим емоційним станом є важливим для нашого особистого та професійного розвитку. З нетерпінням чекаю на ваші думки та обговорення цієї захоплюючої теми.

1.1.1 Комплекс почуттів

Дійсно, емоційний стан включає в себе різні почуття, які можуть бути сильними або слабкими, тривалими або короткочасними. Давайте розглянемо декілька основних емоцій та їх фізіологічні особливості:

Радість: Це позитивний емоційний стан, який викликає почуття щастя та задоволення. Фізіологічно, радість може викликати збільшення серцебиття, посмішку на обличчі та відчуття легкості.

Смуток: Це негативний емоційний стан, який викликає почуття горя та втрати. Фізіологічно, смуток може викликати зниження енергії, поганий апетит та збільшення сну.

Страх: Це негативний емоційний стан, який викликає почуття тривоги та небезпеки. Фізіологічно, страх може викликати збільшення серцебиття, потіння та адреналіну в крові.

Гнів: Це негативний емоційний стан, який викликає почуття роздратування та незадоволення. Фізіологічно, гнів може викликати збільшення серцебиття, напруження м’язів та збільшення адреналіну в крові.

Здивування: Це емоційний стан, який викликає почуття несподіванки та непередбачуваності. Фізіологічно, здивування може викликати широко відкриті очі, збільшення серцебиття та відчуття втрати контролю.

Відраза: Це негативний емоційний стан, який викликає почуття відштовхування та неприємності. Фізіологічно, відраза може викликати відчуття нудоти, зморщення обличчя та відчуття відштовхування.

Ці емоції та їх фізіологічні реакції допомагають нам реагувати на різні ситуації та події в нашому житті. Вони відіграють важливу роль у нашому спілкуванні та взаємодії з іншими людьми. З нетерпінням чекаю на ваші думки та обговорення цієї захоплюючої теми (Рис. 1.1.1).

Рисунок 1.1.1 Емоції людини

1.1.2 Відношення до себе, до інших людей, до подій і предметів

Абсолютно вірно. Наш емоційний стан відображає наше відношення до різних аспектів нашого життя:

Відношення до себе: Наш емоційний стан може відображати наше самопочуття та самооцінку. Наприклад, якщо ми відчуваємо задоволення або гордість, це може вказувати на позитивне ставлення до себе. З іншого боку, почуття вини або сорому можуть відображати негативне ставлення до себе.

Відношення до інших людей: Наші емоції можуть відображати наше ставлення до інших людей. Наприклад, почуття любові або дружби відображають позитивне ставлення до інших, тоді як почуття злості або ненависті відображають негативне ставлення.

Відношення до подій: Наш емоційний стан може відображати наше ставлення до подій у нашому житті. Наприклад, радість або ексайтмент можуть відображати позитивне ставлення до певної події, тоді як смуток або розчарування можуть відображати негативне ставлення.

Відношення до предметів або ситуацій: Наш емоційний стан також може відображати наше ставлення до різних предметів або ситуацій. Наприклад, захоплення або цікавість можуть відображати позитивне ставлення, тоді як страх або відраза можуть відображати негативне ставлення.

Розуміння цих відносин допомагає нам краще розуміти наші власні емоції та емоції інших людей, що може поліпшити наше спілкування та взаємодію з іншими. З нетерпінням чекаю на ваші думки та обговорення цієї захоплюючої теми.

1.1.3 Позитивний, негативний або нейтральний

Емоційний стан може бути позитивним (наприклад, радість, захоплення), негативним (наприклад, гнів, смуток) або нейтральним (коли ви не відчуваєте сильних емоцій).

Позитивним: Це стан, коли ви відчуваєте приємні емоції, такі як радість, захоплення, щастя, задоволення, любов, надія тощо. Позитивні емоції зазвичай

виникають, коли ми отримуємо те, чого хотіли, або коли нас щось приємно здивувало.

Негативним: Це стан, коли ви відчуваєте неприємні емоції, такі як гнів, смуток, страх, відраза, заздрість, ненависть тощо. Негативні емоції зазвичай виникають, коли ми стикаємося з проблемами, конфліктами або розчаруваннями.

Нейтральним: Це стан, коли ви не відчуваєте сильних емоцій. Ви можете відчувати спокій, рівновагу або байдужість. Нейтральний емоційний стан зазвичай виникає, коли ми відчуваємо себе комфортно і нічого нас особливо не турбує.

Розуміння свого емоційного стану допомагає нам краще управляти нашими реакціями на різні ситуації, покращує наше самопочуття та допомагає нам ефективніше взаємодіяти з іншими людьми. З нетерпінням чекаю на ваші думки та обговорення цієї захоплюючої теми.

1.1.4 Вплив на сприйняття, мислення, поведінку та взаємодію

Наш емоційний стан може значно впливати на те, як ми сприймаємо світ навколо нас, як ми думаємо, як ми поводимося, а також на те, як ми взаємодіємо з іншими людьми. Наприклад, коли ми щасливі, ми можемо сприймати світ більш позитивно, мати більш оптимістичні думки, бути більш активними та дружелюбними. З іншого боку, коли ми сумні, ми можемо сприймати світ більш негативно, мати більш песимістичні думки, бути менш активними та відчуженими.

Вплив на сприйняття: Наш емоційний стан може впливати на те, як ми сприймаємо події, людей та ситуації навколо нас. Наприклад, коли ми щасливі, ми можемо сприймати світ більш позитивно, бачити “світло в кінці тунелю” у викликах та бути більш відкритими до нових можливостей.

Вплив на мислення: Наш емоційний стан може впливати на наші думки та рішення. Наприклад, коли ми сумні, ми можемо мати більш песимістичні думки, бути більш критичними до себе та інших, а також бути менш відкритими до нових ідей.

Вплив на поведінку: Наш емоційний стан може впливати на нашу поведінку. Наприклад, коли ми злий, ми можемо бути більш агресивними, коли ми раді, ми

можемо бути більш енергійними та активними, а коли ми сумні, ми можемо бути менш активними та відчуженими.

Вплив на взаємодію: Наш емоційний стан може впливати на те, як ми взаємодіємо з іншими. Наприклад, коли ми щасливі, ми можемо бути більш дружелюбними та відкритими, коли ми злий, ми можемо бути більш відчуженими та агресивними.

Розуміння цих впливів може допомогти нам краще управляти нашими емоціями та покращити наше самопочуття та взаємодію з іншими. З нетерпінням чекаю на ваші думки та обговорення цієї захоплюючої теми.

Таким чином, емоційний стан є важливою частиною нашого життя, яка впливає на багато аспектів нашого буття. Важливо вміти розпізнавати та керувати своїми емоціями, щоб покращити своє самопочуття та взаємодію з іншими.

1.1.5 Темперамент людини

Темперамент — це вроджена стійка властивість людської психіки, яка визначає реакцію людини на інших людей та події, що з нею відбуваються. Існують чотири основних типи темпераменту (Рис. 1.1.5) :

Холерик:

- Висока активність і швидка реакція.
- Енергійний, запальний, може різко змінювати свій емоційний стан.
- Проявляє високу працездатність, але пориви вистачає ненадовго.

Сангвінік:

- Активний і емоційний.
- Позитивний настрій, жвава міміка.
- Легко перемикає увагу, працює над завданнями, які цікаві.

Флегматик:

- Спокійний і врівноважений.
- Незворушний, врівноважений, витривалий.
- Ідеальний виконавець для монотонних завдань.

Меланхолік:

- Чутливий, вразливий, інтроверт.
- Реагує гостро на зовнішні чинники, схильний до смутку.
- Боязкий, виснажується легко.

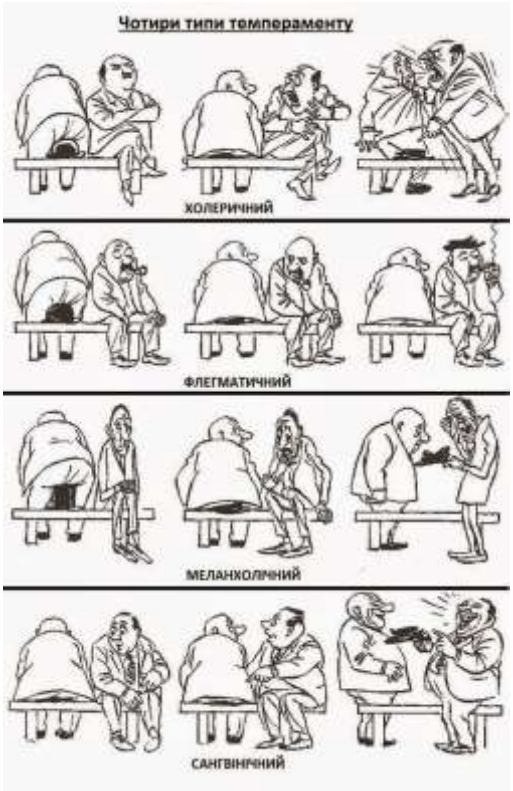


Рисунок 1.1.5 Темперамент людини

1.2 Аналіз тексту

Аналіз тексту - це важливий процес, який допомагає перетворити великі обсяги неструктурованих текстових даних в структуровану форму, яка може бути легко аналізована. Давайте розглянемо детальніше кожен з цих методів:

Токенізація : Це процес розбиття тексту на окремі слова або “токени”. Наприклад, речення “Я люблю програмувати” може бути розбите на токени: “Я”, “люблю”, “програмувати”.

Видалення зайвих символів: Включає в себе видалення непотрібних символів, таких як знаки пунктуації, числа, спеціальні символи тощо.

Видалення стоп-слів: Стоп-слова - це загальновживані слова, які зазвичай не несуть важливого значення, такі як “і”, “а”, “в”, “на” тощо. Вони часто видаляються, щоб зменшити розмір даних та зосередитися на важливих словах.

Стемінг: Це процес перетворення слова до його основної форми. Наприклад, слова “бігає”, “біг” та “бігав” можуть бути перетворені до їх основної форми “біг”.

Лематизація: Це схоже на стемінг, але більш витончений. Лематизація перетворює слово до його кореневої форми, враховуючи контекст. Наприклад, слово “біг” може бути перетворене на “бігти”.

Векторизація тексту: Це процес перетворення тексту в числові вектори, які можуть бути аналізовані алгоритмами машинного навчання. Існує кілька методів векторизації, включаючи Bag of Words, TF-IDF, Word2Vec тощо.

1.2.1 Видалення зайвих символів

Цей процес включає видалення непотрібних символів з тексту, таких як спеціальні символи, цифри, пунктуація тощо. Це допомагає зосередитися на важливих словах у тексті.

Видалення зайвих символів, відоме також як очищення тексту, - це важливий крок у передобробці текстових даних. Давайте розглянемо це більш детально:

Спеціальні символи: Спеціальні символи, такі як @, #, $, %, &, *, (,), _, +, -, =, {, }, [,], |, , :, ;, ', ", <, >, ?, /, ^, ~, `, можуть бути видалені, оскільки вони зазвичай не несуть важливого значення в тексті. Однак вони можуть бути корисними в певних контекстах, наприклад, при аналізі соціальних медіа, де символи @ та # мають специфічне значення.

Цифри: Цифри також можуть бути видалені, особливо якщо вони не несуть важливого значення в контексті аналізу. Однак вони можуть бути корисними при аналізі тексту, що містить важливі числові дані.

Пунктуація: Знаки пунктуації, такі як коми, крапки, двокрапки, крапки з комою, знаки оклику, знаки питання, квадратні та круглі дужки, можуть бути видалені. Вони зазвичай не несуть важливого значення для аналізу тексту, але можуть бути корисними для визначення границь речень або фраз.

Пробіли та табуляція: Надлишкові пробіли, табуляція та переноси рядків також можуть бути видалені, оскільки вони не несуть важливого значення для аналізу тексту.

1.2.2 Стемінг

Стемінг - це процес видалення суфіксів зі слів, щоб отримати їх “стему” або корінь. Наприклад, стемінг слова “running” дасть “run”. Це допомагає згрупувати разом різні форми одного й того ж слова.

Спрощення слів: Стемінг допомагає зменшити слово до його базової форми, що спрощує аналіз тексту. Наприклад, слова “running”, “runner”, “runs” всі зводяться до базової форми “run” (Рис. 1.2.2).

Групування слів: Стемінг допомагає групувати разом різні форми одного й того ж слова. Це полегшує аналіз тексту, оскільки слова з різними формами, але з однаковим коренем, будуть розглядатися як одне й те ж слово.

Швидкість та ефективність: Стемінг може зробити процес аналізу тексту швидшим та ефективнішим, оскільки він зменшує кількість унікальних слів у тексті.

Однак важливо зауважити, що стемінг може не завжди бути ідеальним для всіх задач обробки тексту. Він може призвести до “перестемінгу”, коли слова зводяться до неправильних коренів, або “недостатнього стемінгу”, коли слова не зводяться достатньо. В таких випадках, може бути корисним використовувати більш складні техніки, такі як лематизація. З нетерпінням чекаю на ваші думки та обговорення цієї захоплюючої теми.

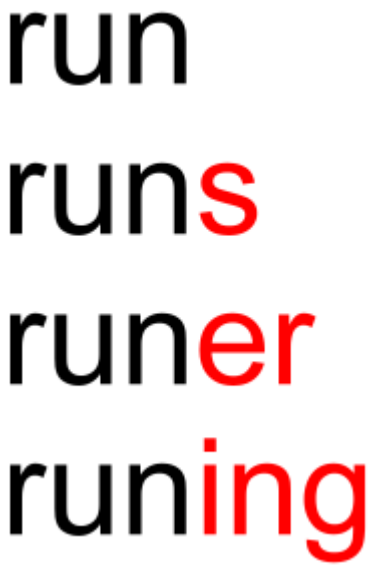


Рисунок 1.2.2 Стемінг

1.2.3 Лематизація

Лематизація - це більш складний процес, ніж стемінг, який перетворює слово на його базову форму або “лему”. Наприклад, лема слова “ran” буде “run”. Відмінність від стемінгу полягає в тому, що лематизація враховує контекст і частину мови слова.

Врахування контексту: Лематизація враховує контекст, в якому використовується слово. Наприклад, слово “біг” може бути перетворене на “бігти”, якщо воно використовується як дієслово, але залишиться незмінним, якщо воно використовується як іменник.

Врахування частини мови: Лематизація також враховує частину мови слова. Наприклад, слово “кращий” може бути перетворене на “добрий”, оскільки воно є прикметником у вищому ступені.

Точність: Лематизація зазвичай дає більш точні результати, ніж стемінг. Наприклад, стемінг слова “гуси” дасть “гус”, тоді як лематизація правильно поверне “гуска”.

Складність: Лематизація - це більш складний процес, ніж стемінг, оскільки вона вимагає глибшого розуміння мови. Вона зазвичай використовує складні алгоритми та великі мовні ресурси, такі як словники та бази даних морфологічних форм.

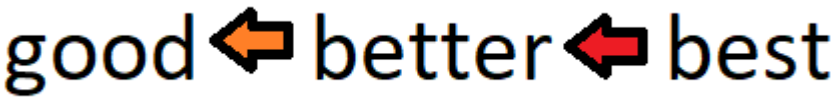


Рисунок 1.2.3 Лематизація

1.2.4 Видалення стоп-слів

Стоп-слова - це загальновживані слова, які не несуть багато значення в тексті, такі як “and”, “the”, “is” тощо. Видалення цих слів допомагає зосередитися на важливих словах у тексті.

Видалення стоп-слів допомагає зменшити розмір даних для обробки та зосередитися на словах, які несуть важливе значення. Це особливо корисно в задачах, пов’язаних з обробкою природної мови (Natural language processing NLP),

таких як аналіз тексту, машинне навчання, майнінг тексту, витягування інформації та інше.

Однак важливо зауважити, що список стоп-слів може варіюватися в залежності від конкретної задачі або додатку. Наприклад, при аналізі відгуків користувачів слово “не” може бути важливим, навіть якщо воно є стоп-словом, оскільки воно може змінити загальний сенс відгуку. З нетерпінням чекаю на ваші думки та обговорення цієї захоплюючої теми.

Ці методи, разом з іншими, допомагають перетворити великі обсяги текстових даних в структуровану форму, яка може бути легко аналізована для отримання цінних висновків. Зауважте, що використання кожного з цих методів залежить від конкретної задачі аналізу тексту.

1.3 Природня мова та його компоненти

Обробка природної мови (Natural language processing NLP) - це дійсно захоплююча галузь штучного інтелекту, яка зосереджена на взаємодії між комп’ютерами та людською мовою. Ось декілька ключових аспектів NLP (Natural language processing)

Синтаксичний аналіз: Це процес визначення структури речень та відношень між словами. Синтаксичний аналіз включає в себе розбиття тексту на фрази, речення та інші компоненти, а також визначення ролей окремих слів у реченні.

Визначення значення слів: Семантичний аналіз включає в себе визначення значення слів на основі їх контексту. Наприклад, слово “летюча миша” може означати тварину, але в контексті комп’ютерних ігор, це може бути назвою персонажа.

Визначення загального значення речення: Семантичний аналіз також включає в себе визначення загального значення речення. Це включає в себе розуміння, як слова взаємодіють одне з одним у реченні, та як ці взаємодії впливають на загальне значення речення.

Врахування семантичних відносин: Семантичний аналіз також включає в себе врахування семантичних відносин між словами, таких як синоніми (слова з

однаковим або подібним значенням), антоніми (слова з протилежним значенням), гіпоніми (слова, які є підмножиною більш загального слова) та інше.

Розбиття тексту: Синтаксичний аналіз починається з розбиття тексту на менші частини, такі як фрази та речення. Це допомагає зрозуміти структуру тексту та визначити, як слова взаємодіють одне з одним.

Розбиття на речення: Текст розбивається на окремі речення, використовуючи знаки пунктуації, такі як крапки, знаки оклику та знаки питання. Це допомагає визначити границі між окремими думками або ідеями в тексті.

Розбиття на слова: Кожне речення подальше розбивається на окремі слова або “токени”. Це зазвичай відбувається за допомогою пробілів, хоча іноді можуть використовуватися інші символи.

Визначення структури речення: Після розбиття тексту на слова можна визначити структуру речення та взаємозв’язки між словами. Наприклад, можна визначити, яке слово є суб’єктом, яке є дієсловом, а яке є об’єктом.

Визначення ролей слів: Кожне слово в реченні має певну роль, наприклад, суб’єкт, дієслово, об’єкт, прикметник тощо. Синтаксичний аналіз допомагає визначити ці ролі, що полегшує розуміння значення речення.

Суб’єкт: Суб’єкт - це основний діяч у реченні. Це той, хто виконує дію або про кого йдеться в реченні. Наприклад, в реченні “Марія читає книгу”, “Марія” є суб’єктом.

Дієслово: Дієслово вказує на дію, яка відбувається в реченні. Воно вказує на те, що робить суб’єкт. Наприклад, в реченні “Марія читає книгу”, “читає” є дієсловом.

Об’єкт: Об’єкт - це те, що прямо взаємодіє з дієсловом. Це те, до чого або до кого спрямована дія. Наприклад, в реченні “Марія читає книгу”, “книгу” є об’єктом.

Прикметник: Прикметник - це слово, яке описує або модифікує іменник або займенник, надаючи додаткову інформацію про нього. Наприклад, в реченні “Марія читає стару книгу”, “стару” є прикметником, який описує “книгу”.

Побудова дерева залежностей: На основі цих ролей можна побудувати дерево залежностей, яке відображає структуру речення та показує, як слова взаємодіють одне з одним.

Дерево залежностей - це графічне представлення структури речення, яке відображає граматичні відносини між словами. Ось декілька ключових аспектів побудови дерева залежностей:

Вузли та ребра: У дереві залежностей кожне слово в реченні представляється як вузол, а граматичні відносини між словами представляються як ребра, що з'єднують вузли.

Напрямок залежностей: Ребра в дереві залежностей мають напрямок. Наприклад, у реченні "Марія читає книгу", ребро від "читає" до "Марія" вказує на те, що "Марія" є суб'єктом дієслова "читає".

Типи залежностей: Кожне ребро в дереві залежностей має певний тип, який відображає граматичну роль, яку відіграє слово в реченні. Наприклад, у реченні "Марія читає книгу", ребро від "читає" до "Марія" може мати тип "nsubj" (номінативний суб'єкт), а ребро від "читає" до "книгу" - тип "obj" (об'єкт).

Використання дерева залежностей: Дерева залежностей можуть бути використані для багатьох задач обробки природної мови, включаючи синтаксичний аналіз, визначення частин мови, визначення семантичних ролей, машинний переклад та інше.

Використання граматичних правил: Синтаксичний аналіз також включає в себе використання граматичних правил мови для перевірки правильності речення та визначення його структури.

Розповідне речення:

Це речення, яке передає інформацію про події, факти або стани. Воно описує те, що відбувається, відбулося або буде відбуватися. Приклад: "Сонце сходить на сході."

У розповідних реченнях використовуються звичайні форми дієслів.

Їх можна визначити за наявністю підмета та присудка.

Питальне речення:

Це речення, яке містить запитання. Воно використовується для отримання інформації або вираження цікавості.

Питальні речення починаються з дієслова або дієслівного словосполучення.

Приклад: “Чи їли ви сьогодні сніданок?”

Спонукальне речення:

Це речення, яке спонукує до дії. Воно використовується для наказів, прохань, закликів тощо.

Спонукальні речення можуть починатися з дієслова у повелительному способі або з інших словосполучень, які виражають спонукання до дії.

Приклад: “Зачекайте, будь ласка.”

1.3.1 Обробка природної мови (Natural language processing NLP)

NLP (Natural language processing) використовує різні техніки для аналізу тексту, включаючи синтаксичний аналіз (визначення граматичної структури речень), семантичний аналіз (розуміння значення), та дискурсивний аналіз (розуміння контексту).

Синтаксичний аналіз: Це процес визначення граматичної структури речень. Синтаксичний аналіз включає в себе розбиття тексту на фрази, речення та інші компоненти, а також визначення ролей окремих слів у реченні.

Семантичний аналіз: Це процес визначення значення слів та речень. Семантичний аналіз включає в себе визначення значення слів на основі їх контексту та визначення загального значення речення.

Дискурсивний аналіз: Це процес визначення, як контекст впливає на значення речення. Дискурсивний аналіз включає в себе врахування культурних та соціальних факторів при інтерпретації тексту.

1.3.2 Розуміння природної мови (Natural language understanding NLU)

Розуміння природної мови (Natural language understanding NLU) - це технічна концепція в рамках ширшої теми обробки природної мови. NLU (Natural language understanding) - це процес перекладу природної людської мови у форму, яку може інтерпретувати комп'ютер. По суті, перш ніж комп'ютер зможе обробляти

лінгвістичні дані, він повинен зрозуміти ці дані. Це важливий аспект розвитку штучного інтелекту, що дозволяє системам взаємодіяти з користувачами на більш природному рівні, розуміючи їхні запити і відповіді.

Історія розуміння природної мови рясніє відкриттями і досягненнями. Ще в 1960-х роках були спроби використовувати комп'ютери для розуміння природної мови. Програма STUDENT, написана Даніелем Бобровим, показала, як комп'ютери можуть розуміти прості дані на природній мові та розв'язувати алгебраїчні задачі. Згодом з'явилися такі системи, як ELIZA і SHRDLU, які відіграли важливу роль у розвитку галузі.

Серед застосувань розуміння природної мови - автоматизовані міркування, машинний переклад, системи запитань і відповідей, збір новин, голосова активація та багато інших. Це допомагає покращити взаємодію між людиною та комп'ютером, зробити її більш ефективною та зручною.

1.3.3 Генерація природної мови (Natural language generation NLG)

Генерація природної мови — це процес, в якому комп’ютери перетворюють структуровані дані на текст, який звучить природно для людей. Це може включати створення відповідей в чат-ботах, написання статей, генерацію поезії, автоматичне створення оголошень та багато інших застосувань. NLG (Natural language generation) дозволяє комп’ютерам взаємодіяти з нами за допомогою тексту, що робить комунікацію більш природною та зрозумілою.

1.3.4 Оцінка природної мови (Natural Language Assessment NLA)

Оцінка природної мови (Natural Language Assessment NLA) — це процес вимірювання та оцінювання здібностей систем обробки природної мови в розумінні, аналізі, генерації та інтерпретації тексту. Метою NLA є визначення ефективності алгоритмів та моделей обробки природної мови, виявлення їх сильних та слабких сторін, а також встановлення стандартів для порівняння різних підходів.

Оцінка природної мови включає кілька ключових аспектів:

- Точність (Accuracy): Вимірюється, наскільки правильно система обробки природної мови виконує завдання, такі як розпізнавання частин мови, синтаксичний аналіз, семантичне розпізнавання та інші.
- Повнота (Completeness): Оцінюється, наскільки повно система охоплює всі аспекти заданого завдання. Наприклад, при аналізі тексту важливо, щоб система не пропустила жодної значущої деталі.
- Продуктивність (Efficiency): Визначається швидкість і ресурсомісткість системи при виконанні завдань обробки природної мови. Це особливо важливо для реальних застосувань, де швидкість відповіді є критичною.
- Зрозумілість (Interpretability): Оцінюється, наскільки легко люди можуть зрозуміти та інтерпретувати результати роботи системи. Це важливо для застосувань, де потрібна взаємодія людини з системою або коли результати повинні бути прозорими для користувачів.
- Робастність (Robustness): Оцінюється здатність системи обробляти непередбачувані або незвичайні вхідні дані без значних збоїв або помилок.
- Для проведення оцінки природної мови використовуються різні методи та інструменти, включаючи тестові набори даних, змагальні задачі, користувацькі дослідження та автоматизовані метрики, такі як BLEU (Bilingual Evaluation Understudy) для машинного перекладу або ROUGE (Recall-Oriented Understudy for Gisting Evaluation) для оцінки якості резюмування тексту.

Оцінка природної мови є важливою складовою розробки та впровадження систем обробки природної мови, оскільки вона допомагає дослідникам і розробникам вдосконалювати моделі та алгоритми, забезпечуючи високий рівень якості та надійності кінцевих продуктів.

1.3.5 Машинний переклад (МП)

Машинний переклад (МП) — це процес, за допомогою якого комп’ютери автоматично перекладають текст з однієї мови на іншу. Ось деякі ключові аспекти машинного перекладу:

Автоматизація перекладу:

Машинний переклад дозволяє замінити ручний переклад, що збільшує продуктивність та знижує витрати часу.

Комп’ютери використовують алгоритми та моделі, щоб перетворити текст з однієї мови на іншу.

Моделі машинного навчання:

Машинний переклад використовує моделі машинного навчання, такі як нейронні мережі, щоб вивчати зв’язки між словами та фразами в різних мовах.

Ці моделі навчаються на великих корпусах текстів, щоб вивчити синтаксичні та семантичні правила.

Типи машинного перекладу:

Словниковий переклад: Використовує попередньо складений словник для перекладу слів.

Статистичний переклад: Аналізує статистику великих текстових корпусів для визначення найбільш ймовірних перекладів.

Нейронний переклад: Використовує нейронні мережі для більш точного перекладу, враховуючи контекст.

Виклики машинного перекладу:

Амбігвітність: Деякі слова можуть мати кілька значень, і вибір правильного перекладу залежить від контексту.

Культурні відмінності: Різні мови мають різні вирази та культурні особливості, які можуть впливати на переклад.

Постійне вдосконалення:

Машинний переклад постійно вдосконалюється завдяки новим даним та технологіям.

Людська редакція залишається важливою для досягнення найкращого результату.

1.3.6 Розпізнавання мови

Розпізнавання мови — це технологія, яка дозволяє комп’ютерам аналізувати, розуміти та інтерпретувати людську мову. Ось деякі ключові аспекти розпізнавання мови:

Розпізнавання голосу:

Системи розпізнавання голосу перетворюють звукові сигнали (голосові команди, розмови) на текст.

Це використовується в голосових помічниках, системах автоматичного відповідання та інших додатках.

Обробка природної мови:

NLP дозволяє аналізувати та розуміти текстові дані, включаючи граматику, семантику та контекст.

Використовується для створення голосових помічників, перекладу мов, аналізу тексту та багатьох інших завдань.

Машинний переклад:

Це одна з найвідоміших застосувань NLP, яка дозволяє комп’ютерам автоматично перекладати текст з однієї мови на іншу.

Використовується в онлайн-перекладачах, програмах для міжнародних комунікацій та бізнесу.

Голосові помічники:

Голосові помічники, такі як Siri, Alexa, Google Assistant, Cortana використовують NLP для розпізнавання голосових команд та надання відповідей.

Виклики та покращення:

Розпізнавання мови має свої виклики, такі як амбігвітність, культурні відмінності та точність.

Постійне вдосконалення технологій NLP допомагає зробити взаємодію з комп’ютерами більш природною та зручною.NLP відкриває безліч можливостей для взаємодії між людьми та комп’ютерами, від простого перекладу тексту до

складного аналізу емоцій та відтворення людської мови. Це дуже захоплююча область, яка продовжує розвиватися.

1.4 Машинне навчання

Машинне навчання — це розділ штучного інтелекту, який дозволяє комп’ютерним системам навчатися на даних і покращувати свої результати без явного програмування (не потрібно писати інструкції для кожної задачі). Це важливий інструмент, який змінює спосіб, яким ми вирішуємо завдання, відповідаємо на запитання та аналізуємо дані.

Машинне навчання може бути застосовано в різних сферах, включаючи аналіз тексту, класифікацію об’єктів, прогнозування, рекомендації, обробку зображень та багато інших. У контексті аналізу емоційного стану, машинне навчання може використовуватися для класифікації тексту на основі емоційного контексту. Наприклад, визначення, чи текст виражає радість, сум, обурення або інші емоції.

2 ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ

В цьому розділі ми розглянемо деякі ключові технології та інструменти, які використовуються для розробки веб-сайтів:

- HTML (Hypertext Markup Language): HTML є стандартною мовою розмітки для створення веб-сторінок. Використовується для визначення структури та вмісту веб-документів.
- CSS (Cascading Style Sheets): CSS використовується для стилізації веб-сторінок, включаючи кольори, шрифти, розташування елементів тощо.
- JavaScript: JavaScript — це мова програмування, яка дозволяє додавати динаміку та інтерактивність на веб-сторінки. Використовується для створення взаємодії з користувачем, анімації, перевірки форм тощо.
- JSON (JavaScript Object Notation): JSON (JavaScript Object Notation) - це легкий та гнучкий формат обміну даними, який використовується для представлення структурованих даних. Він базується на тексті та використовує зрозумілу для людей структуру, що робить його простим для читання та розуміння.
- SQL (Structured Query Language) - це мова програмування, яка використовується для керування та вилучення даних з реляційних баз даних. Ці бази даних організовують дані в таблиці з чітко визначеними зв'язками між ними.
- Python: Python — це високорівнева мова програмування, яка використовується для розробки веб-додатків, обробки даних, машинного навчання та багатьох інших завдань.
- JQuery: JQuery — це бібліотека JavaScript, яка спрощує взаємодію з DOM (Document Object Model) та робить розробку більш зручною.
- Bootstrap : Bootstrap — це найновіша версія популярного фреймворку для створення відзивчивих, мобільних веб-сайтів.Включає в себе готові компоненти, стилізацію та JavaScript-плагіни.

2.1 HTML: HyperText Markup Language «мова гіпертекстової текстової розмітки»

HTML (HyperText Markup Language) - це мова розмітки гіпертексту, яка використовується для створення структури та вмісту веб-сторінок. Вона не є мовою програмування,але використовується браузерами для інтерпретації та візуального представлення вмісту веб-сторінок. Уявіть собі HTML як план будівлі.Він описує різні частини будівлі та те, як вони взаємопов'язані між собою.

Ось деякі з ключових характеристик HTML:

- Простота: HTML має простий синтаксис, що робить його легким для вивчення та використання.
- Стандарт: HTML є стандартизованою мовою, що означає, що веб-сторінки, написані HTML, будуть однаково відображатися в різних браузерах.
- Гнучкість: HTML можна використовувати для створення різних типів веб-сторінок, від простих статичних сторінок до складних динамічних веб-додатків.
- Доступність: HTML підтримує доступність, що робить веб-сторінки доступними для людей з обмеженими можливостями.
- Широка підтримка: HTML широко підтримується всіма сучасними браузерами.

Як HTML використовується для створення веб-сторінок:

- Елементи: HTML використовує теги для визначення різних елементів веб-сторінки, таких як заголовки, абзаци, списки, зображення та посилання.
- Атрибути: Теги HTML можуть мати атрибути, які надають додаткову інформацію про елемент.
- Структура: Елементи HTML організовані в ієрархічну структуру, яка визначає макет веб-сторінки.
- Вміст: Вміст HTML може включати текст, зображення, відео, код та інші дані.

Приклад простого HTML-документа:

```
<!DOCTYPE html>
<html>
<head>
   <title>Моя перша веб-сторінка</title>
</head>
<body>
   <h1>Заголовок</h1>
   <p>Це абзац тексту.</p>
   <img src="image.jpg" alt="Опис зображення">
   <a href="https://www.example.com">Посилання</a>
</body>
</html>
```

HTML - це основа для створення веб-сторінок. Вивчивши HTML, ви зможете створювати власні веб-сайти та веб-додатки але так було не завжди.

2.1.1 HTML 1.0

HTML 1.0, випущений у 1991 році, був першою офіційною версією мови розмітки HTML. Він заклав фундамент для сучасної веб-розробки, але з часом став застарілим і був замінений новішими версіями, такими як HTML 2.0, HTML 4.01 та HTML5.

Структура HTML 1.0:

Документ HTML 1.0 мав просту структуру, що складалася з двох основних секцій:

- <HEAD>: Ця секція містила метадані про веб-сторінку, такі як заголовок, опис та посилання на зовнішні ресурси.
- <BODY>: Ця секція містила видимий вміст веб-сторінки, включаючи текст, зображення, посилання та інші елементи.

Елементи HTML 1.0:

HTML 1.0 містив обмежений набір елементів, які використовувалися для структурування та форматування контенту. До деяких поширених елементів

належали:

- <h1> - <h6>: Визначали заголовки різного рівня (від найбільшого до найменшого).
- <p>: Визначав абзац тексту.
-
: Вставляв розрив рядка.
- <img>: Вставляв зображення.
- <a>: Визначав гіперпосилання.
- : Визначав несортований список.
- : Визначав сортований список.
- : Визначав елемент списку.
- <table>: Визначав таблицю.
- <tr>: Визначав рядок таблиці.
- <td>: Визначав комірку таблиці.

Атрибути HTML 1.0:

Елементи HTML 1.0 могли мати атрибути, які надавали додаткову інформацію про них. Наприклад, елемент <a> міг мати атрибут href, який визначав URL-адресу гіперпосилання.

Обмеження HTML 1.0:

HTML 1.0 був досить обмеженою мовою, що робило його не таким гнучким, як сучасні версії HTML. Деякі з його обмежень включали:

- Відсутність підтримки CSS, що ускладнювало форматування веб-сторінок.
- Відсутність підтримки JavaScript, що ускладнювало додавання динамічності до веб-сторінок.
- Відсутність підтримки багатьох сучасних елементів HTML, таких як відео та аудіо.

Незважаючи на свої обмеження, HTML 1.0 відіграв важливу роль у розвитку Всесвітньої павутини. Він заклав фундамент для сучасних веб-технологій і допоміг зробити Інтернет доступним для широкої аудиторії.

2.1.2 HTML 2.0

HTML 2.0: Опис

HTML 2.0, випущений у 1995 році, став значним кроком вперед у порівнянні з HTML 1.0. Він запровадив ряд нових функцій, які зробили веб-сторінки більш динамічними та інтерактивними.

Нові функції HTML 2.0:

- Підтримка CSS: HTML 2.0 дозволив використовувати CSS для форматування веб-сторінок, що значно полегшило контроль над зовнішнім виглядом сторінок.
- Підтримка JavaScript: HTML 2.0 дозволив використовувати JavaScript для додавання динамічності до веб-сторінок, що зробило їх більш інтерактивними.
- Нові елементи: HTML 2.0 додав ряд нових елементів, таких як <form>, <input>, <select> та <textarea>, які полегшили створення веб-форм.
- Покращена підтримка таблиць: HTML 2.0 покращив підтримку таблиць, додавши нові атрибути та елементи.
- Підтримка фреймів: HTML 2.0 дозволив використовувати фрейми для розбиття веб-сторінки на декілька частин.

Обмеження HTML 2.0:

Незважаючи на свої переваги, HTML 2.0 все ще мав деякі обмеження:

- Відсутність чіткої стандартизації: Специфікації HTML 2.0 не були чітко стандартизовані, що призвело до несумісності між браузерами.
- Надмірне використання тегів: HTML 2.0 часто використовувався надмірно, що призвело до нечіткого та складного коду.
- Відсутність підтримки доступності: HTML 2.0 не мав вбудованої підтримки доступності, що ускладнювало людям з інвалідністю доступ до веб-сторінок.

Незважаючи на свої обмеження, HTML 2.0 відіграв важливу роль у розвитку Всесвітньої павутини.

2.1.3 HTML 3.0

HTML 3.0: Опис

HTML 3.0, випущений у 1995 році, був порівняно невеликим оновленням HTML 2.0.

Він запровадив кілька нових функцій, але загалом не мав значного впливу на веб-розробку.

Нові функції HTML 3.0:

- Нові елементи: HTML 3.0 додав кілька нових елементів, таких як <applet>, <dir>, <menu> та <sub>/<sup>.
- Покращена підтримка таблиць: HTML 3.0 покращив підтримку таблиць, додавши нові атрибути та елементи.
- Підтримка математичних формул: HTML 3.0 додав підтримку математичних формул за допомогою елемента <math>.

Обмеження HTML 3.0:

HTML 3.0 мав багато тих самих обмежень, що й HTML 2.0, включаючи:

- Відсутність чіткої стандартизації: Специфікації HTML 3.0 не були чітко стандартизовані, що призвело до несумісності між браузерами.
- Надмірне використання тегів: HTML 3.0 часто використовувався надмірно, що призвело до нечіткого та складного коду.
- Відсутність підтримки доступності: HTML 3.0 не мав вбудованої підтримки доступності, що ускладнювало людям з інвалідністю доступ до веб-сторінок.

Через свої обмеження HTML 3.0 швидко був замінений HTML 4.0.

2.1.4 HTML 4.0

HTML 4.0, випущений у 1998 році, став значним оновленням HTML 2.0 і HTML 3.0. Він запровадив ряд нових функцій, які зробили веб-сторінки більш структурованими, доступними та сумісними з різними браузерами.

Нові функції HTML 4.0:

- Строгі та перехідні режими: HTML 4.0 запровадив два режими: строгий та перехідний. Строгий режим відповідав новим стандартам W3C, тоді як перехідний режим дозволяв використовувати деякі застарілі елементи та атрибути з HTML 2.0 і HTML 3.0.
- Покращена підтримка CSS: HTML 4.0 покращив підтримку CSS, що зробило його більш гнучким для форматування веб-сторінок.
- Нові елементи: HTML 4.0 додав ряд нових елементів, таких як <object>, <embed>, <abbr>,<acronym>, <cite>, <dfn>, <kbd>, <samp>, , <var>, <ins>, <del>, <pre>,<q>, <sub>, <sup>, <address>, <blockquote>, <dd>, <dl>, <dt>, <fieldset>,<form>, <frame>, <frameset>, <h1 - <h6>>, <hr>, <iframe>, <img>, <input>,<kbd>, <label>, <legend>, <li>, <link>, <map>, <menu>, <meta>, <nobr>,<ol>, <optgroup>, <option>, <p>, <param>, <pre>, <q>, <rb>, <rt>, <rp>,<ruby>, <s>, <samp>, <script>, <select>, , <srclang>, ,<style>, <sub>, <sup>, <table>, <tbody>, <td>, <th>, <thead>, <title>,<tr>, <ul>, <var>, <wbr>, які дозволили створювати більш структурований та доступний контент.
- Покращена підтримка таблиць: HTML 4.0 покращив підтримку таблиць, додавши нові атрибути та елементи.
- Підтримка XML: HTML 4.0 додав підтримку XML, що дозволило вставляти XML-дані на веб-сторінки.
- Доступність: HTML 4.0 мав вбудовану підтримку доступності, що полегшило людям з інвалідністю доступ до веб-сторінок.

Переваги HTML 4.0:

- Покращена структура: HTML 4.0 зробив веб-сторінки більш структурованими, що покращило їх доступність та зрозумілість для пошукових систем.
- Більш доступний: HTML 4.0 мав вбудовану підтримку доступності, що полегшило людям з інвалідністю доступ до веб-сторінок.
- Сумісність з браузерами: HTML 4.0 був добре сумісний з різними браузерами.

HTML 4.0 був широко використаний для створення веб-сайтів наприкінці 1990-х та на початку 2000-х років.

З часом його було замінено HTML5.0, який пропонує більше можливостей та кращу підтримку сучасних веб-технологій.

2.1.5 HTML 5.0

HTML5.0, випущений у 2014 році, став найбільшим оновленням мови розмітки HTML за багато років.

Він запровадив безліч нових функцій, які зробили веб-розробку більш потужною, гнучкою та доступною.

Нові функції HTML5.0:

- Нові семантичні елементи: HTML5.0 додав безліч нових семантичних елементів, таких як <header>, <nav>, <section>, <article>, <aside>, <footer>, <figure>, <figcaption>, <details>, <summary>, <time>, <mark>, <ruby>, <rt>, <rp>, <data>, <datalist>, <meter>, <progress>, <canvas>, <video>, <audio>, <source>, <track>, <embed>, <iframe>, <map>, <area>, <command>, <search>, <menuitem>, <details>, <summary>, <time>, <mark>, <ruby>, <rt>, <rp>, <data>, <datalist>, <meter>, <progress>, <canvas>, <video>, <audio>, <source>, <track>, <embed>, <iframe>, <map>, <area>, <command>, <search>, <menuitem>, які дозволяють описувати контент більш чітко та зрозуміло.
- Підтримка мультимедіа: HTML5.0 додав вбудовану підтримку мультимедіа, що дозволяє вставляти відео та аудіо на веб-сторінки без використання сторонніх плагінів.
- API веб-форм: HTML5.0 додав API веб-форм, що полегшує роботу з веб-формами та збирання даних користувачів.
- Локальне сховище: HTML5.0 додав локальне сховище, яке дозволяє веб-сайтам зберігати дані на комп'ютері користувача.
- WebSockets: HTML5.0 додав підтримку WebSockets, що дозволяє веб-сайтам встановлювати двосторонні з'єднання з серверами.

- MathML: HTML5.0 додав підтримку MathML, що дозволяє вставляти математичні формули на веб-сторінки.
- Drag and drop: HTML5.0 додав підтримку drag and drop, що дозволяє користувачам перетягувати файли та інші елементи на веб-сторінки.
- Geolocation: HTML5.0 додав підтримку geolocation, що дозволяє веб-сайтам отримувати доступ до географічного розташування користувача.
- Canvas: HTML5.0 додав елемент <canvas>, який дозволяє малювати графіку на веб-сторінках за допомогою JavaScript.
- SVG: HTML5.0 додав підтримку SVG, що дозволяє вставляти масштабовані векторні зображення на веб-сторінки.

Переваги HTML5.0:

- Покращена семантика: HTML5.0 робить веб-сторінки більш семантичними, що покращує їх зрозумілість для пошукових систем та допоміжних технологій.
- Більш гнучкий: HTML5.0 пропонує більше можливостей для створення динамічних та інтерактивних веб-сторінок.
- Легший у вивченні: HTML5.0 має більш чіткий та зрозумілий синтаксис, що робить його легшим для вивчення.
- Підтримка сучасних браузерів: HTML5.0 підтримується всіма сучасними браузерами.

HTML5.0 став стандартом для веб-розробки і використовується для створенняширокого спектру веб-сайтів та веб-додатків.

2.2 CSS Cascading Style Sheets «каскадні таблиці стилів»

CSS (Cascading Style Sheets) - це мова опису зовнішнього вигляду веб-сторінок. Вона використовується для форматування HTML-документів, контролюючи такі аспекти, як:

- Кольори: Визначення кольору тексту, фону та інших елементів сторінки.
- Шрифти: Вибір шрифтів, розміру шрифту та ваги тексту.

- Розташування: Визначення розташування елементів на сторінці, таких як заголовки, абзаци та зображення.
- Відступи та облямівки: Додавання відступів і облямівок навколо елементів для контролю їх пробілу.

Декорації: Додавання декорацій, таких як рамки, тіні та градієнти. Переваги використання CSS:

- Покращення візуального оформлення: CSS дозволяє створювати більш привабливі та зручні для користувачів веб-сторінки.
- Відділення вмісту від стилю: CSS відокремлює опис вмісту сторінки (HTML) від її візуального оформлення (CSS), що робить код більш чітким і легким для підтримки.
- Повторне використання стилів: CSS дозволяє повторно використовувати стилі на різних сторінках сайту, що економить час і код.
- Покращення доступності: CSS можна використовувати для покращення доступності веб-сторінок для людей з обмеженими можливостями.

Приклади використання CSS:

- Зміна кольору тексту заголовків на синій.
- Використання шрифту Arial для всього тексту на сторінці.
- Розташування зображення зліва від тексту.
- Додавання рамки навколо таблиці.
- Створення кнопки з градієнтом та тінню.

Загалом, CSS - це важливий інструмент для веб-розробників, який дозволяє створювати більш привабливі, зручні та доступні веб-сторінки.

2.2.1 CSS 1.0

CSS 1.0: Опис

CSS 1.0, випущений у 1996 році, був першою офіційною версією каскадної таблиці стилів (CSS). Він заклав фундамент для сучасного веб-дизайну, але з часом став застарілим і був замінений новішими версіями, такими як CSS 2.0, CSS 3.0 та CSS4.

Можливості CSS 1.0:CSS 1.0 пропонував обмежений набір можливостей для форматування веб-сторінок.

До його основних функцій належали:

- Селектори: CSS 1.0 дозволяв використовувати селектори для вибору елементів HTML, які потрібно було стилізувати.
- Властивості: CSS 1.0 дозволяв використовувати властивості для визначення різних аспектів зовнішнього вигляду елементів, таких як колір, шрифт, вирівнювання та розміри.
- Значення властивостей: CSS 1.0 пропонував обмежений набір значень властивостей, таких як ключові слова та числові значення.
- Спадщина: CSS 1.0 підтримував спадщину, що дозволяло властивостям автоматично застосовуватися до дочірніх елементів.

Обмеження CSS 1.0:

CSS 1.0 був досить обмеженою мовою, що робило його не таким гнучким, як сучасні версії CSS.

Деякі з його обмежень включали:

- Відсутність позиціонування: CSS 1.0 не мав можливості позиціонувати елементи на веб-сторінці.
- Відсутність макетів: CSS 1.0 не мав можливості створювати складні макети веб-сторінок.
- Обмежена підтримка шрифтів: CSS 1.0 пропонував обмежену підтримку шрифтів, що ускладнювало використання спеціальних шрифтів на веб-сторінках.
- Відсутність підтримки анімації: CSS 1.0 не мав можливості створювати анімації на веб-сторінках.

Незважаючи на свої обмеження, CSS 1.0 відіграв важливу роль у розвитку Всесвітньої павутини.Він заклав фундамент для сучасного веб-дизайну і допоміг зробити веб-сторінки більш візуально привабливими.

2.2.2 CSS 2.0

CSS 2.0, випущений у 1998 році, став значним оновленням CSS 1.0. Він запровадив ряд нових функцій, які зробили веб-дизайн більш гнучким та потужним.

Нові функції CSS 2.0:

- Позиціонування: CSS 2.0 додав підтримку позиціонування, що дозволило розміщувати елементи на веб-сторінці більш точно.
- Макеты: CSS 2.0 додав підтримку макетів, що дозволило створювати більш складні макети веб-сторінок.
- Медіа-типи: CSS 2.0 додав підтримку медіа-типів, що дозволило застосовувати різні стилі до веб-сторінок залежно від типу пристрою, на якому вони переглядаються.
- Модулі: CSS 2.0 був розділений на модулі, що зробило його більш організованим та легшим для вивчення.
- Нові властивості: CSS 2.0 додав ряд нових властивостей, таких як властивості для роботи зі шрифтами, таблицями та кордонами.
- Псевдокласи: CSS 2.0 додав псевдокласи, які дозволили застосовувати стилі до елементів залежно від їхнього стану.

Переваги CSS 2.0:

- Більш гнучкий: CSS 2.0 зробив веб-дизайн більш гнучким, завдяки можливостям позиціонування, макетів та медіа-типів.
- Більш потужний: CSS 2.0 надав більше можливостей для контролю над зовнішнім виглядом веб-сторінок.
- Легший у вивченні: Завдяки модульній структурі CSS 2.0 стало легше вивчати.

CSS 2.0 широко використовувався для створення веб-сайтів наприкінці 1990-х та на початку 2000-х років.З часом його було замінено CSS 3.0, який пропонує ще більше можливостей та кращу підтримку сучасних веб-технологій.

2.2.3 CSS 3.0

CSS 3.0, випущений у 2009 році, став революційним оновленням каскадної таблиці стилів (CSS). Він запровадив безліч нових функцій, які зробили веб-дизайн більш гнучким, динамічним та візуально вражаючим.

Нові функції CSS 3.0:

- Модульна структура: CSS 3.0 має модульну структуру, що робить його більш організованим та легшим для вивчення.
- Нові селектори: CSS 3.0 додав ряд нових селекторів, які дозволили більш точно вибирати елементи для стилізації.
- Нові властивості: CSS 3.0 додав безліч нових властивостей, які стосуються різних аспектів зовнішнього вигляду веб-сторінок, таких як градієнти, тіні, заокруглення кутів, анімації, переходи, трансформації та багато іншого.
- Медіа-запити: CSS 3.0 розширив можливості медіа-запитів, що дозволило застосовувати різні стилі до веб-сторінок залежно від розміру екрана, орієнтації пристрою та інших факторів.
- Шрифти: CSS 3.0 додав кращу підтримку шрифтів, що дозволило використовувати веб-шрифти та вбудовувати шрифти на веб-сторінки.
- Макеты: CSS 3.0 додав нові можливості для створення макетів веб-сторінок, таких як flexbox та grid layout.
- Анімації: CSS 3.0 додав кращу підтримку анімацій, що дозволило створювати більш складні та динамічні анімації на веб-сторінках.
- Переходи: CSS 3.0 додав підтримку переходів, що дозволило створювати плавні переходи між різними станами елементів.

Переваги CSS 3.0:

- Більш гнучкий: CSS 3.0 робить веб-дизайн більш гнучким завдяки новим селекторам, властивостям та можливостям макетів.
- Більш динамічний: CSS 3.0 дозволяє створювати більш динамічні та інтерактивні веб-сторінки за допомогою анімацій та переходів.

- Більш візуально вражаючий: CSS 3.0 пропонує широкий спектр можливостей для створення візуально вражаючих веб-сторінок за допомогою градієнтів, тіней, заокруглення кутів та інших ефектів.
- Краща підтримка сучасних браузерів: CSS 3.0 добре підтримується сучасними браузерами.

CSS 4.0 все ще перебуває в розробці, але він обіцяє ще більше можливостей для веб-дизайну, таких як 3D-розмітка, інтерактивні елементи та краща підтримка голосових інтерфейсів.

2.3 JavaScript «ECMAScript»

ES (EcmaScript) - це мова програмування, яка використовується для створення динамічних веб-сторінок та веб-додатків. Вона є основою JavaScript і постійно розвивається з випуском нових версій з новими функціями та можливостями.

Ось розшифровка та опис деяких поширених назв версій ES:

- ES3: Це перша широко розповсюджена версія ES, випущена в 1999 році. Вона має обмежений набір функцій і зараз не рекомендується для використання.
- ES5: Ця версія, випущена в 2009 році, додала багато нових функцій та можливостей, які стали основою сучасного JavaScript.
- ES6 (ES2015): Ця значна версія, випущена в 2015 році, ввела багато нових синтаксичних форм та функцій, які значно розширили можливості JavaScript.
- ES7 (ES2016): Ця версія, випущена в 2016 році, додала кілька нових функцій, таких як оператори ... та async/await.
- ES8 (ES2017): Ця версія, випущена в 2017 році, додала кілька нових функцій, таких як об'єкти Object.values() та Object.entries().
- ES9 (ES2018): Ця версія, випущена в 2018 році, додала кілька нових функцій, таких як Promise.allSettled() та Symbol.toStringTag().

- ES10 (ES2019): Ця версія, випущена в 2019 році, додала кілька нових функцій, таких як Array.prototype.flat() та Object.prototype.fromEntries().

Важливо зазначити, це те що:

- ES3, ES4, ES5 та ES9 офіційно не називалися ES*. Ці назви використовувалися спільнотою JavaScript для позначення цих версій.
- ES6, ES7, ES8, ES10 - це офіційні назви, визначені ECMA International.
- ES2015, ES2016, ES2017, ES2018, ES2019 - це альтернативні назви для ES6, ES7, ES8, ES9 та ES10 відповідно.

Загалом, ES постійно розвивається, додаючи нові функції та можливості, які роблять JavaScript потужнішим та зручнішим для використання.

2.3.1 JSON «JavaScript Object Notation»

JSON (JavaScript Object Notation) - це текстовий формат обміну даними, легкий для читання людьми та машинами. Він використовується для зберігання та передачі структурованих даних між веб-додатками, веб-серверами та іншими системами.

Ось деякі з ключових характеристик JSON:

- Простота: JSON має простий синтаксис, заснований на тексті, що робить його легким для розуміння та використання.
- Гнучкість: JSON може використовуватися для зберігання різних типів даних, таких як числа, рядки, масиви та об'єкти.
- Стандарт: JSON є стандартизованим форматом, що означає, що його можна легко обробляти різними мовами програмування.
- Незалежність від мови: JSON не залежить від жодної мови програмування, що робить його універсальним форматом для обміну даними.
- Широка підтримка: JSON широко підтримується всіма сучасними веб-браузерами та мовами програмування.

Структура JSON:

- Дані JSON організовані у пари ключ-значення.
- Ключі - це рядки, що описують дані.

- Значеннями можуть бути числа, рядки, масиви, об'єкти або значення null.
- Масиви - це впорядковані колекції значень.
- Об'єкти - це не впорядковані колекції пар ключ-значення.

Приклад JSON-об'єкта, що описує користувача:

Приклад простого JSON-документа

```
{
  "name": "Якусевич Максим",
  "age": 30,
  "city": "Київ",
  "skills": ["програмування", "адміністрування", "навчання"],
  "isAdmin": true
}
```

JSON - це зручний та універсальний формат для обміну даними, що робить його популярним вибором для веб-застосунків та API.

2.4 SQL

SQL (Structured Query Language) - це мова декларативного програмування, що використовується для взаємодії з реляційними базами даних. Вона дозволяє керувати даними в базі даних, такими як додавання, видалення, оновлення та вибірка.

Ось деякі з ключових характеристик SQL:

- Простота: SQL має простий синтаксис, що робить його легким для вивчення та використання.
- Потужність: SQL дає можливість виконувати складні операції з даними в базі даних.
- Стандарт: SQL є стандартизованим мовою, що означає, що його можна використовувати з різними системами керування базами даних (СКБД).

- Гнучкість: SQL можна використовувати для різних завдань, таких як створення таблиць, вставка даних, вибірка даних, оновлення даних та видалення даних.
- Широка підтримка: SQL широко підтримується всіма основними СКБД.

Основні команди SQL:

- SELECT: Використовується для вибірки даних з таблиць.
- INSERT: Використовується для вставки даних у таблиці.
- UPDATE: Використовується для оновлення даних у таблицях.
- DELETE: Використовується для видалення даних з таблиць.
- CREATE TABLE: Використовується для створення нових таблиць.
- DROP TABLE: Використовується для видалення таблиць.
- Приклад SQL-запиту для вибірки всіх імен та прізвищ користувачів з таблиці users:

SQL

SELECT name, surname FROM users;

SQL - це незамінний інструмент для будь-кого, хто працює з реляційними базами даних.

2.5 JQuery

jQuery: Короткий опис

jQuery - це легка бібліотека JavaScript, яка спрощує роботу з HTML, CSS та JavaScript. Вона робить роботу з DOM (Document Object Model) простішою та зручнішою, завдяки чому ви можете швидше створювати динамічні та інтерактивні веб-сторінки.

Ось деякі з ключових переваг використання jQuery:

- Простота: jQuery має простий синтаксис, що робить її легкою для вивчення та використання.
- Гнучкість: jQuery можна використовувати для різних завдань, таких як маніпулювання DOM, обробка подій, анімація та AJAX.

- Широка підтримка: jQuery широко підтримується та використовується на мільйонах веб-сайтів.
- Багато плагінів: Існує безліч плагінів jQuery, які розширюють її можливості та додають нові функції.
- Ось деякі з основних можливостей jQuery:
- Вибір елементів: jQuery надає зручні методи для вибору елементів DOM за різними критеріями.
- Маніпулювання DOM: jQuery дозволяє легко змінювати вміст, атрибути та стилі елементів DOM.
- Обробка подій: jQuery спрощує обробку подій, таких як кліки, зміни тексту та прокрутка.
- Анімація: jQuery надає методи для створення плавних та динамічних анімацій.
- AJAX: jQuery полегшує асинхронні запити до сервера та оновлення сторінки без перезавантаження.

Приклад використання jQuery для вибору всіх елементів p на сторінціта зміни їх кольору на червоний:

JavaScript - файл

```
$(document).ready(function() {
  $("p").css("color", "red");
});
```

jQuery - це потужний інструмент, який може значно полегшити розробку веб-сайтів JavaScript.

2.6 Bootstrap

Bootstrap - це безкоштовний фреймворк HTML, CSS та JavaScript, який використовується для створення адаптивних та мобільних веб-сайтів. Він полегшує розробку веб-сайтів з елегантним дизайном та зручним користувацьким інтерфейсом, які добре виглядають на будь-якому пристрої.

Ось деякі з ключових переваг використання Bootstrap:

- Простота: Bootstrap має простий синтаксис та чітку структуру, що робить його легким для вивчення та використання.
- Адаптивність: Bootstrap створює веб-сайти, які автоматично адаптуються до різних розмірів екранів та пристроїв, завдяки чому вони добре виглядають на комп'ютерах, планшетах та смартфонах.
- Мобільність: Bootstrap спеціально розроблений для створення мобільних веб-сайтів, які забезпечують оптимальний користувацький досвід на смартфонах та планшетах.
- Готові компоненти: Bootstrap містить набір готових компонентів, таких як кнопки, форми, таблиці, навігаційні панелі та модальні вікна, які економлять час та спрощують розробку.
- Безкоштовний та відкритий: Bootstrap є безкоштовним та відкритим з вихідним кодом, що робить його доступним для будь-кого.

Ось деякі з основних компонентів Bootstrap:

- Сітка: Система сітки Bootstrap дозволяє легко розбивати макет веб-сторінки на ряди та стовпці.
- Типографіка: Bootstrap надає стилі для заголовків, абзаців, списків та іншого текстового контенту.
- Форми: Bootstrap містить готові компоненти для створення різних типів форм, таких як текстові поля, кнопки радіо та прапорці.
- Кнопки: Bootstrap надає стилі для різних типів кнопок з різними кольорами та розмірами.
- Таблиці: Bootstrap містить стилі для створення чітких та зручних таблиць.
- Алерти: Bootstrap надає компоненти для відображення повідомлень про помилки, попередження та успішні дії.
- Модальні вікна: Bootstrap дозволяє створювати модальні вікна для відображення додаткового контенту або запитів користувача.

Приклад використання Bootstrap для створення простої кнопки:

Приклад простого HTML-класу з використанням Bootstrap

<button class="btn btn-primary">Моя кнопка</button>

Bootstrap - це потужний інструмент, який може значно полегшити розробку веб-сайтів.

2.6.1 Bootstrap 1

Bootstrap 1, випущений у 2011 році, був першою версією популярного фреймворку HTML, CSS та JavaScript для створення веб-сайтів.

Він відіграв важливу роль у ранньому розвитку адаптивного веб-дизайну, але з часом став застарілим і був замінений новішими версіями, такими як Bootstrap 2, 3, 4 та 5.

Ось деякі з ключових характеристик Bootstrap 1:

- 12-стовпцева сітка: Bootstrap 1 використовував 12-стовпцеву сітку для створення макетів веб-сторінок. Це дозволяло веб-розробникам створювати адаптивні веб-сайти, які добре виглядали на різних розмірах екранів.
- Компоненти: Bootstrap 1 пропонував ряд попередньо створених компонентів, таких як кнопки, форми, таблиці та навігаційні панелі. Це економило час веб-розробникам і гарантувало, що їхні веб-сайти мали послідовний дизайн.
- Стилі: Bootstrap 1 поставлявся з попередньо визначеними стилями CSS, які можна використовувати для стилізації веб-сайтів. Це економило час веб-розробникам і гарантувало, що їхні веб-сайти мали чистий та сучасний дизайн.
- Підтримка JavaScript: Bootstrap 1 включав плагіни JavaScript, які можна використовувати для додавання інтерактивності до веб-сайтів, таких як розкривні меню та каруселі.

Відмінності Bootstrap 1 від сучасних версій:

Bootstrap 1 значно відрізняється від сучасних версій фреймворку в декількох аспектах:

- Дизайн: Дизайн Bootstrap 1 був більш простим і мінімалістичним, ніж у сучасних версіях.

- Компоненти: Сучасні версії Bootstrap пропонують набагато ширший спектр компонентів, ніж Bootstrap 1.
- Стилі: Стилі CSS у сучасних версіях Bootstrap більш модульні та гнучкі, ніж у Bootstrap 1.
- Підтримка браузерів: Сучасні версії Bootstrap підтримують ширший спектр браузерів, ніж Bootstrap 1.
- Доступність: Сучасні версії Bootstrap приділяють більше уваги доступності, ніж Bootstrap 1.

Хоча Bootstrap 1 більше не використовується, він відіграв важливу роль у розвитку адаптивного веб-дизайну.Сучасні версії Bootstrap ґрунтуються на його фундаменті і пропонують набагато ширший спектр функцій та можливостей.

2.6.2 Bootstrap 2

Bootstrap 2, випущений у 2012 році, був наступником Bootstrap 1 і значно розширив можливості фреймворку.Він став ще більш популярним, допомогаючи веб-розробникам створювати адаптивні та мобільні веб-сайти.

Ось деякі з ключових характеристик Bootstrap 2:

- Вдосконалена сітка: Сітка Bootstrap 2 була більш гнучкою, ніж у Bootstrap 1, з можливістю використовувати до 12 стовпців на рядку або різну кількість стовпців у різних рядках.
- Більше компонентів: Bootstrap 2 пропонував ширший спектр компонентів, ніж Bootstrap 1, включаючи елементи навігації, форми, таблиці, модальні вікна та багато іншого.
- Вдосконалені стилі: Стилі CSS у Bootstrap 2 були більш модульними та гнучкими, ніж у Bootstrap 1, що дозволяло веб-розробникам легше налаштовувати дизайн своїх веб-сайтів.
- Підтримка JavaScript: Bootstrap 2 включав більше плагінів JavaScript, ніж Bootstrap 1, додаючи більше інтерактивності до веб-сайтів.

- Підтримка адаптивних компонентів: Bootstrap 2 включав адаптивні компоненти, які автоматично змінювали свій розмір та розташування залежно від розміру екрана.

Відмінності Bootstrap 2 від сучасних версій:

Bootstrap 2 значно відрізняється від сучасних версій фреймворку в декількох аспектах:

- Дизайн: Дизайн Bootstrap 2 був більш простим і мінімалістичним, ніж у сучасних версіях.
- Компоненти: Сучасні версії Bootstrap пропонують набагато ширший спектр компонентів, ніж Bootstrap 2.
- Стилі: Стилі CSS у сучасних версіях Bootstrap більш модульні та гнучкі, ніж у Bootstrap 2.
- Підтримка браузерів: Сучасні версії Bootstrap підтримують ширший спектр браузерів, ніж Bootstrap 2.
- Доступність: Сучасні версії Bootstrap приділяють більше уваги доступності, ніж Bootstrap 2.

Хоча Bootstrap 2 більше не використовується, він відіграв важливу роль у розвитку адаптивного веб-дизайну. Сучасні версії Bootstrap ґрунтуються на його фундаменті і пропонують набагато ширший спектр функцій та можливостей.

2.6.3 Bootstrap 3

Bootstrap 3, випущений у 2013 році,став значною модернізацією фреймворку, заклавши основу для його сучасного вигляду та функціональності.

Він здобув широку популярність завдяки своїй гнучкості,адаптивності та зручності використання.

Ось деякі з ключових характеристик Bootstrap 3:

- Вдосконалена сітка: Сітка Bootstrap 3 залишилась 12-стовпцевою, але стала більш гнучкою, дозволяючи використовувати різну кількість стовпців у різних рядках та адаптуватися до різних розмірів екранів.

- Оновлені компоненти: Bootstrap 3 пропонував оновлені та розширені компоненти, такі як кнопки, форми, таблиці, навігаційні панелі, модальні вікна та багато іншого.
- Flat Design: Bootstrap 3 впровадив Flat Design, що дало йому більш сучасний та мінімалістичний вигляд.
- Підтримка LESS та SASS: Bootstrap 3 додавав підтримку LESS та SASS, полегшуючи розробникам налаштування стилів.
- Реагуючі компоненти: Bootstrap 3 включав адаптивні компоненти, які автоматично змінювали свій розмір та розташування залежно від розміру екрана.

Відмінності Bootstrap 3 від сучасних версій:

Bootstrap 3 значно відрізняється від сучасних версій фреймворку в декількох аспектах:

- Дизайн: Bootstrap 3 мав більш Flat Design, ніж сучасні версії, які використовують більш плавні та округлені елементи.
- Компоненти: Сучасні версії Bootstrap пропонують набагато ширший спектр компонентів, ніж Bootstrap 3.
- Стилі: Стилі CSS у сучасних версіях Bootstrap більш модульні та гнучкі, ніж у Bootstrap 3.
- Підтримка браузерів: Сучасні версії Bootstrap підтримують ширший спектр браузерів, ніж Bootstrap 3.
- Доступність: Сучасні версії Bootstrap приділяють більше уваги доступності, ніж Bootstrap 3.

Хоча Bootstrap 3 більше не є офіційно підтримуваним, він все ще широко використовується багатьма веб-сайтами.Сучасні версії Bootstrap ґрунтуються на його фундаменті і пропонують набагато ширший спектр функцій та можливостей.

2.6.4 Bootstrap 4

Bootstrap 4, випущений у 2018 році, був значним оновленням фреймворку, яке модернізувало його дизайн, додало нові функції та покращило доступність. Він

став ще більш популярним, допомогаючи веб-розробникам створювати сучасні та адаптивні веб-сайти.

Ось деякі з ключових характеристик Bootstrap 4:

- Оновлений дизайн: Bootstrap 4 використовує більш плавні та округлені елементи, а також покращену типографію та кольорову палітру.
- Нові компоненти: Bootstrap 4 додає нові компоненти, такі як картки, групи кнопок, а також покращені версії існуючих компонентів.
- Підтримка Flexbox: Bootstrap 4 використовує Flexbox для макетування, що дає веб-розробникам більше гнучкості при створенні складних макетів.
- Підтримка Grid: Bootstrap 4 зберігає 12-стовпцеву сітку, але також пропонує нову систему сітки Grid, яка дає більше можливостей для макетування.
- Покращена доступність: Bootstrap 4 приділяє більше уваги доступності, з покращеною підтримкою ARIA та іншими функціями.

Відмінності Bootstrap 4 від сучасних версій:

Bootstrap 4 значно відрізняється від сучасних версій фреймворку в декількох аспектах:

- Дизайн: Bootstrap 4 має більш сучасний дизайн, ніж сучасні версії, які використовують Material Design.
- Компоненти: Сучасні версії Bootstrap пропонують набагато ширший спектр компонентів, ніж Bootstrap 4.
- Стилі: Стилі CSS у сучасних версіях Bootstrap більш модульні та гнучкі, ніж у Bootstrap 4.
- Підтримка браузерів: Сучасні версії Bootstrap підтримують ширший спектр браузерів, ніж Bootstrap 4.
- Доступність: Сучасні версії Bootstrap приділяють більше уваги доступності, ніж Bootstrap 4.

Хоча Bootstrap 4 більше не оновлюється, він все ще широко використовується багатьма веб-сайтами.Сучасні версії Bootstrap ґрунтуються на його фундаменті і пропонують набагато ширший спектр функцій та можливостей.

2.7 Python

Python - це універсальна мова програмування високого рівня, яка використовується в широкому спектрі задач, як-от веб-розробка, машинне навчання, аналітика даних та наукові обчислення. Вона відома завдяки своїй простоті, читабельності та елегантності, що робить її чудовим вибором як для початківців, так і для досвідчених розробників.

Ось деякі з ключових характеристик Python:

- Простота: Python має простий синтаксис, подібний до англійської мови, що робить його легким для вивчення та використання.
- Читабельність: Код Python легко читати та розуміти, що робить його зручним для співпраці та обслуговування.
- Універсальність: Python можна використовувати для різних задач, від веб-розробки до машинного навчання та наукових обчислень.
- Багато бібліотек: Python має обширну екосистему бібліотек та фреймворків, які підтримують широкий спектр задач.
- Активна спільнота: Python має велику та активну спільноту розробників, які готові допомогти та надати підтримку.

Деякі з застосувань Python:

- Веб-розробка: Python можна використовувати для створення веб-сайтів та веб-додатків за допомогою фреймворків, таких як Django та Flask.
- Машинне навчання: Python є популярною мовою для машинного навчання завдяки бібліотекам, таким як TensorFlow та scikit-learn.
- Аналітика даних: Python часто використовується для аналітики даних та візуалізації за допомогою бібліотек, таких як Pandas та Matplotlib.
- Наукові обчислення: Python можна використовувати для наукових обчислень та чисельних методів за допомогою бібліотек, таких як NumPy та SciPy.
- Автоматизація: Python можна використовувати для автоматизації задач та створення сценаріїв за допомогою модуля subprocess.

Приклад простого коду Pythonдля друку привітання:

Python

print("Привіт, світе!")

Python - це потужна та універсальна мова програмування, яка може бути корисною в багатьох сферах.

2.7.1 Python 1

Python 1 - це перша версія мови програмування Python, яка була випущена в 1991 році. Вона відіграла важливу роль у ранньому розвитку мови і заклала фундамент для її сучасних версій.Однак Python 1 більше не підтримується з 2020 року і не рекомендується для нових проектів.

Ось деякі з ключових характеристик Python 1:

- Простий синтаксис: Python 1 використовує простий та зрозумілий синтаксис, який робить його легким для вивчення та використання.
- Інтерпретована мова: Python 1 - це інтерпретована мова, що означає, що код виконується рядок за рядком. Це робить його гнучким та динамічним.
- Підтримка об'єктно-орієнтованого програмування: Python 1 підтримує об'єктно-орієнтоване програмування (ООР), що дозволяє розробникам створювати модульні та масштабовані програми.
- Багата стандартна бібліотека: Python 1 поставляється з багатою стандартною бібліотекою, яка містить модулі для поширених завдань, таких як робота з файлами, мережеві комунікації та обробка рядків.

Відмінності Python 1 від сучасних версій:

Python 1 значно відрізняється від сучасних версій мовив декількох аспектах:

- Підтримка: Python 1 більше не підтримується з 2020 року, що означає, що для нього не випускаються оновлення безпеки або виправлення помилок.
- Функції: Сучасні версії Python додають багато нових функцій та можливостей, яких не було в Python 1. Це стосується розуміння списків, форматування рядків, операторів та багато іншого.
- Бібліотеки: Сучасні версії Python мають ширший спектр доступних бібліотек та фреймворків, які роблять Python більш потужним та універсальним.

- Швидкість: Сучасні версії Python зазвичай працюють швидше, ніж Python 1.

Хоча Python 1 більше не використовується,він відіграв важливу роль у розвитку мови. Сучасні версії Python ґрунтуються на його фундаменті і пропонують набагато ширший спектр функцій та можливостей.

2.7.2 Python 2

Python 2 - це друга основна версія мови програмування Python, яка була випущена в 2000 році. Вона стала дуже популярною мовою для веб-розробки, наукових обчислень та машинного навчання. Однак Python 2 більше не підтримується з 2020 року і не рекомендується для нових проектів.

Ось деякі з ключових характеристик Python 2:

- Простий синтаксис: Python 2 використовує простий та зрозумілий синтаксис, який робить його легким для вивчення та використання.
- Інтерпретована мова: Python 2 - це інтерпретована мова, що означає, що код виконується рядок за рядком. Це робить його гнучким та динамічним.
- Підтримка об'єктно-орієнтованого програмування: Python 2 підтримує об'єктно-орієнтоване програмування (ООР), що дозволяє розробникам створювати модульні та масштабовані програми.
- Багата стандартна бібліотека: Python 2 поставляється з багатою стандартною бібліотекою, яка містить модулі для поширених завдань, таких як робота з файлами, мережеві комунікації та обробка рядків.

Відмінності Python 2 від сучасних версій:

Python 2 значно відрізняється від сучасних версій мови в декількох аспектах:

- Підтримка: Python 2 більше не підтримується з 2020 року, що означає, що для нього не випускаються оновлення безпеки або виправлення помилок.
- Функції: Сучасні версії Python додають багато нових функцій та можливостей, яких не було в Python 2. Це стосується розуміння списків, форматування рядків, операторів та багато іншого.

- Бібліотеки: Сучасні версії Python мають ширший спектр доступних бібліотек та фреймворків, які роблять Python більш потужним та універсальним.
- Швидкість: Сучасні версії Python зазвичай працюють швидше, ніж Python 2.

Хоча Python 2 більше не використовується, він відіграв важливу роль у розвитку мови. Сучасні версії Python ґрунтуються на його фундаменті і пропонують набагато ширший спектр функцій та можливостей.

2.7.3 Python 3

Python 3 - це сучасна та активно підтримувана версія мови програмування Python. Вона була випущена в 2008 році і здобула широку популярність завдяки своїй гнучкості, читабельності, потужності та широкому спектру застосувань.

Ось деякі з ключових характеристик Python 3:

- Простий синтаксис: Python 3 використовує простий та зрозумілий синтаксис, який робить його легким для вивчення та використання.
- Інтерпретована мова: Python 3 - це інтерпретована мова, що означає, що код виконується рядок за рядком. Це робить його гнучким та динамічним.
- Підтримка об'єктно-орієнтованого програмування: Python 3 підтримує об'єктно-орієнтоване програмування (ООР), що дозволяє розробникам створювати модульні та масштабовані програми.
- Багата стандартна бібліотека: Python 3 поставляється з багатою стандартною бібліотекою, яка містить модулі для поширених завдань, таких як робота з файлами, мережеві комунікації та обробка рядків.
- Підтримка Unicode: Python 3 нативно підтримує Unicode, що робить його зручним для роботи з текстом різними мовами.
- Підтримка нових функцій: Python 3 регулярно оновлюється новими функціями та можливостями, що робить його сучасним та актуальним.

Переваги використання Python 3:

- Швидкий початок роботи: Python 3 пропонує широкий спектр попередньо створених модулів та бібліотек, які можна використовувати для швидкого створення програм.
- Універсальність: Python 3 можна використовувати для широкого спектру задач, включаючи веб-розробку, наукові обчислення, машинне навчання, системне адміністрування та багато іншого.
- Читабельність: Python 3 має простий та лаконічний синтаксис, що робить код легким для читання та розуміння.
- Активна спільнота: Python 3 має велику та активну спільноту розробників, які готові допомогти та надати підтримку.
- Безкоштовний та відкритий код: Python 3 є безкоштовним та відкритим кодом, що робить його доступним для всіх.

Деякі недоліки використання Python 3:

- Несумісність з Python 2: Python 3 несумісний з Python 2, що може ускладнити перехід з однієї версії на іншу, якщо у вас вже є код Python 2.
- Складність для початківців: Python 3 може бути складним для початківців порівняно з деякими іншими мовами програмування через його динамічну природу та систему вводу даних.

Загалом, Python 3 - це потужна, гнучка та універсальна мова програмування, яка підходить для широкого спектру задач. Вона проста у вивченні, має багато функцій та активно підтримується.

2.7.3.1 Бібліотеки в Python: Їхня важливість та переваги

Бібліотеки відіграють надзвичайно важливу роль в екосистемі Python та є необхідними інструментами для будь-якого розробника Python. Вони пропонують широкий спектр готових функцій та модулів, що значно розширюють можливості мови та полегшують вирішення задач.

Ось декілька причин, чому бібліотеки Python такі важливі:

- Економія часу та зусиль: Бібліотеки забезпечують готові рішення для поширених задач, завдяки чому вам не потрібно писати код з нуля. Це економить ваш час та зменшує зусилля, необхідні для розробки програм.
- Повторне використання коду: Бібліотеки дозволяють повторно використовувати код, що покращує продуктивність та зменшує дублювання коду. Ви можете використовувати функціональність з бібліотек у різних проектах, не переписуючи її знову і знову.
- Підвищення продуктивності: Бібліотеки надають оптимізовані та ефективні функції, що покращує продуктивність вашого коду. Це економить час та збільшує масштабованість ваших програм.
- Розширення можливостей: Бібліотеки додають нові функціональні можливості до Python, що робить його більш універсальним та потужним. Ви можете використовувати бібліотеки для машинного навчання, аналітики даних, розробки веб-додатків, наукових обчислень та багато іншого.
- Підтримка спільноти: Багато бібліотек Python мають активні спільноти розробників, які надають підтримку, документацію та приклади коду. Це може бути незамінним ресурсом при виникненні проблем або при вивченні нових функцій.

Наприклад:

- Розробка веб-сайту: Замість того, щоб писати код для обробки HTTP-запитів, шаблонів та сеансів, ви можете використовувати бібліотеку Django або Flask, які полегшують та прискорюють процес розробки.
- Аналітика даних: Замість того, щоб писати код для очищення, аналізу та візуалізації даних, ви можете використовувати бібліотеку Pandas, яка надає потужні інструменти для роботи з даними.
- Машинне навчання: Замість того, щоб писати код для реалізації алгоритмів машинного навчання, ви можете використовувати бібліотеку TensorFlow або scikit-learn, які пропонують готові реалізації та інструменти для машинного навчання.

В цілому, бібліотеки є необхідною частиною екосистеми Python та пропонують безліч переваг для розробників. Вони економлять час, покращують продуктивність, розширюють можливості та спрощують розробку програм. Вивчення та використання бібліотек Python є обов'язковим для будь-якого розробника, який хоче створювати ефективні та надійні програми.

2.7.3.1.1 NLTK: Natural Language Toolkit «інструментарій природної мови»

NLTK, або Natural Language Toolkit, є популярною бібліотекою Python для обробки природної мови (NLP). Вона пропонує широкий спектр інструментів та алгоритмів для роботи з текстовими даними, що робить її цінним інструментом для розробників та дослідників, які прагнуть створити інтелектуальні програми, які можуть розуміти та генерувати людську мову.

Ось деякі з ключових можливостей NLTK:

- Токенізація: Розбивка тексту на окремі слова, речення та інші мовні одиниці.
- Стеммінг та лематизація: Зведення слів до їх основної форми для полегшення порівняння та аналізу.
- Частотний аналіз: Визначення частоти слів та інших мовних одиниць у тексті.
- Частини мови (POS) тегування: Ідентифікація частин мови кожного слова в реченні (наприклад, іменник, дієслово, прикметник).
- Синтаксичний аналіз: Визначення граматичної структури речень.
- Семантичний аналіз: Визначення значення слів та фраз у контексті речень.
- Класифікація тексту: Автоматичне категоризування текстових документів за темами або класами.
- Витяг інформації: Виявлення та вилучення конкретних даних з тексту (наприклад, імен, дат, місць).
- Машинний переклад: Переклад тексту з однієї мови на іншу.

- Переваги використання NLTK:
- Широкий спектр можливостей: NLTK пропонує широкий спектр інструментів для різних задач NLP.
- Безкоштовний та відкритий код: NLTK є безкоштовним та відкритим кодом, що робить його доступним для всіх.
- Велика спільнота: NLTK має велику та активну спільноту розробників та користувачів, які можуть надати підтримку та ресурси.
- Простота використання: NLTK має відносно простий у використанні інтерфейс API, що робить його доступним для розробників з різним рівнем досвіду.
- Інтеграція з іншими бібліотеками: NLTK можна легко інтегрувати з іншими популярними бібліотеками Python, такими як NumPy та Pandas.

Деякі недоліки використання NLTK:

- Крива навчання: NLTK може мати трохи складну криву навчання для початківців, особливо для деяких більш просунутих задач.
- Швидкість: Деякі алгоритми NLTK можуть бути повільними для обробки великих обсягів текстових даних.
- Необхідність в сторонніх бібліотеках: Деякі більш просунуті можливості NLTK можуть потребувати використання сторонніх бібліотек.

Загалом, NLTK - це потужний та універсальний інструмент для обробки природної мови на Python. Він пропонує широкий спектр можливостей для роботи з текстовими даними і є цінним ресурсом для розробників та дослідників, які прагнуть створити інтелектуальні програми, які можуть розуміти та генерувати людську мову.

2.7.3.1.2 Flask

Python Flask - це популярний фреймворк веб-додатків Python, який використовується для створення динамічних веб-сайтів та веб-API. Він пропонує простий та елегантний інтерфейс API, що робить його легким у використанні для розробників з різним рівнем досвіду.

Flask використовується для створення широкого спектру веб-додатків, включаючи:

- Веб-сайти: Блоги, форуми, інтернет-магазини та інші типи веб-сайтів з динамічним контентом.
- Веб-API: Інтерфейси програмування додатків, які дозволяють іншим програмам взаємодіяти з вашими даними та сервісами.
- Адміністративні панелі: Інструменти управління для веб-додатків, які дозволяють користувачам входити в систему та керувати вмістом.
- Односторінкові програми (SPA): Інтерактивні веб-сторінки, які завантажуються динамічно і не потребують перезавантаження сторінки.

Ось деякі з ключових характеристик Flask:

- Простота: Flask має простий та елегантний інтерфейс API, який робить його легким у вивченні та використанні.
- Гнучкість: Flask можна використовувати для створення широкого спектру веб-додатків, від простих веб-сайтів до складних веб-API.
- Розширюваність: Flask можна розширювати за допомогою багатьох сторонніх розширень, які додають нові функції та можливості.
- Легкість тестування: Flask легко тестувати за допомогою інструментів тестування, таких як unittest і pytest.
- Масштабованість: Flask можна використовувати для створення масштабованих веб-додатків, які можуть обробляти велику кількість трафіку.

Переваги використання Flask:

- Швидкий початок роботи: Flask пропонує простий синтаксис та мінімальний набір функцій, що дозволяє розробникам швидко розпочати роботу над створенням веб-додатків.
- Легкість вивчення: Flask має чітку та лаконічну документацію, що робить його легким для вивчення навіть для початківців.
- Гнучкість: Flask можна використовувати для створення різних типів веб-додатків, що робить його універсальним інструментом.

- Велике співтовариство: Flask має велике та активне співтовариство розробників, які можуть надати підтримку та ресурси.

Деякі недоліки використання Flask:

- Відсутність деяких функцій: Flask є відносно легким фреймворком і може не мати деяких функцій, які є в більш складних фреймворках.
- Безпека: Розробникам необхідно вживати заходів безпеки при створенні веб-додатків з Flask.
- Крива навчання: Flask може мати трохи складну криву навчання для початківців, особливо для складних веб-додатків.

Загалом, Python Flask - це потужний та гнучкий фреймворк веб-додатків, який є чудовим вибором для розробників, які прагнуть створювати динамічні та масштабовані веб-сайти. Він простий у вивченні, має чітку документацію і пропонує широкий спектр функцій.

2.7.3.1.3 Jinja

Python Jinja - це популярний шаблонний двигун Python, який використовується для динамічного генерування HTML та іншого веб-контенту. Він пропонує потужний та гнучкий синтаксис, що робить його цінним інструментом для розробників, які прагнуть створювати динамічні та інтерактивні веб-сайти.

Jinja використовується для різних цілей, включаючи:

- Генерація HTML-сторінок: Jinja може генерувати динамічні HTML-сторінки на основі даних, що робить його ідеальним для створення персоналізованих веб-додатків.
- Створення електронних листів: Jinja може генерувати динамічні електронні листи на основі даних, що робить його корисним для маркетингових кампаній та розсилок.
- Розробка чат-ботів: Jinja може генерувати динамічні відповіді для чат-ботів на основі вхідних даних користувача.
- Генерація тексту: Jinja може генерувати динамічний текст на основі даних, що робить його корисним для створення звітів та іншого контенту.

Ось деякі з ключових характеристик Jinja:

- Простота: Jinja має простий та елегантний синтаксис, який робить його легким у вивченні та використанні.
- Потужність: Jinja пропонує широкий спектр функцій для динамічного генерування контенту.
- Гнучкість: Jinja можна використовувати для створення різних типів веб-контенту.
- Розширюваність: Jinja можна розширювати за допомогою багатьох сторонніх розширень, які додають нові функції та можливості.
- Легкість тестування: Jinja легко тестувати за допомогою інструментів тестування, таких як unittest і pytest.

Переваги використання Jinja:

- Швидкий початок роботи: Jinja пропонує простий синтаксис та мінімальний набір функцій, що дозволяє розробникам швидко розпочати роботу з динамічним генеруванням контенту.
- Легкість вивчення: Jinja має чітку та лаконічну документацію, що робить його легким для вивчення навіть для початківців.
- Гнучкість: Jinja можна використовувати для створення різних типів веб-контенту, що робить його універсальним інструментом.
- Велике співтовариство: Jinja має велике та активне співтовариство розробників, які можуть надати підтримку та ресурси.

Деякі недоліки використання Jinja:

- Складність для початківців: Jinja може мати трохи складну криву навчання для початківців, особливо для складних шаблонів.
- Безпека: Розробникам необхідно вживати заходів безпеки при використанні Jinja для генерування динамічного контенту.
- Залежність від Python: Jinja є шаблонним двигуном Python, тому для його використання потрібні знання Python.

Загалом, Python Jinja - це потужний та гнучкий шаблонний двигун, який є чудовим вибором для розробників, які прагнуть створювати динамічні та інтерактивні веб-сайти. Він простий у вивченні, має чітку документацію і пропонує широкий спектр функцій.

2.7.3.1.4 JSON

Python має вбудований модуль json, який робить роботу з JSON

простою та зручною. За допомогою цього модуля ви можете завантажувати JSON-дані з файлів або рядків, перетворювати їх на об'єкти Python та знову перетворювати на JSON для зберігання або передачі. Ось деякі з ключових функцій модуля json:

- json.dumps(): Перетворює об'єкт Python у рядок JSON.
- json.loads(): Перетворює рядок JSON в об'єкт Python.
- json.dump(): Записує об'єкт Python у файл JSON.
- json.load(): Завантажує JSON-дані з файлу в об'єкт Python.

Цей код спочатку завантажує JSON-дані з файлу data.json до змінної data.

Потім він використовує type(data), щоб перевірити тип даних, який є словником Python. Нарешті, він перебирає словник data та друкує кожен ключ і відповідне значення.

Приклад перетворення об'єкта Python на JSON та його збереження у файлі: Python

Цей код створює словник Python data, а потім використовує json.dumps(), щоб перетворити його на рядок JSON. Нарешті, він записує рядок JSON у файл output.json. Це лише два приклади того, як можна використовувати модуль json для роботи з JSON у Python. Модуль пропонує набагато більше функцій для завантаження, обробки та збереження JSON-даних.

2.7.3.1.5 SQLAlchemy

SQLAlchemy - це популярна об'єктно-реляційна прошарок (ORM) для Python,яка полегшує взаємодію з базами даних. Вона дозволяє розробникам

працювати з базами даних через об'єкти Python, замість того, щоб писати складні SQL-запити.

Ось деякі з ключових характеристик SQLAlchemy:

- Простота: SQLAlchemy має простий та інтуїтивно зрозумілий API, що робить її легкою для вивчення та використання.
- Потужність: SQLAlchemy надає широкий спектр функцій для роботи з базами даних, таких як визначення моделей, виконання CRUD-операцій (створення, читання, оновлення та видалення), виконання складних запитів та управління транзакціями.
- Гнучкість: SQLAlchemy можна використовувати з різними типами баз даних, такими як PostgreSQL, MySQL, SQLite та Oracle.
- Продуктивність: SQLAlchemy використовує оптимізовані методи для доступу до даних та мінімізації кількості SQL-запитів, що покращує продуктивність ваших програм.
- Активна спільнота: SQLAlchemy має велику та активну спільноту розробників, які надають підтримку, документацію та приклади коду.

Деякі з застосувань SQLAlchemy:

- Розробка веб-додатків: SQLAlchemy можна використовувати для створення моделей даних та взаємодії з базами даних у веб-застосунках Python за допомогою фреймворків, таких як Django та Flask.
- Аналітика даних: SQLAlchemy можна використовувати для доступу до даних з баз даних та перетворення їх на структури даних Python для аналізу та візуалізації.
- Наукові обчислення: SQLAlchemy можна використовувати для зберігання та завантаження наукових даних з баз даних.
- Автоматизація: SQLAlchemy можна використовувати для автоматизації задач, пов'язаних з базами даних, таких як міграція даних та резервне копіювання.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

В цій частині все треба показувати на очно щоб було зрозуміло що відбувається та передається і накладалось з усіх розділів до цього.

3.1 Інтерфейс

В першу чергу ми перейдемо до створення шаблону який буде створений за допомогою Jinja та накладатися на всі файли будуть наслідувати один HTML файл а контент буде мінятися залежно від запиту. Цей файл називають base.html (Рис. 3.1.1)

Рисунок 3.1.1 Приклад реалізації шаблону сайту

Після цього ми повинні перейти до створення головних сторінок сайтів які будуть наповнюватися інформацією в блоки спеціальні для контенту. Наприклад головна сторінка буде виглядати даним чином (Рис. 3.1.2)

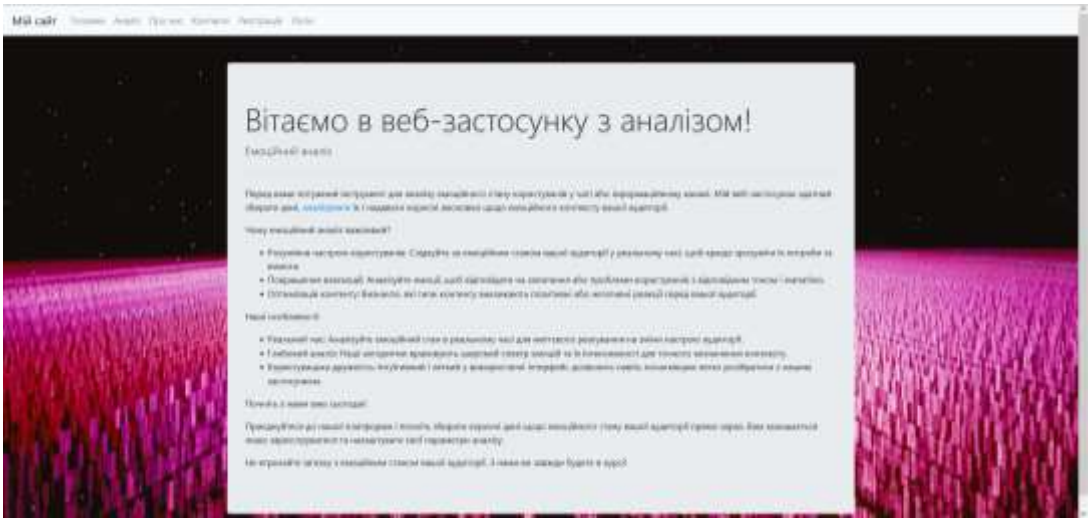


Рисунок 3.1.2 Головна сторінка сайту

Також цим можна побачити сторінку про розробника-здобувача який це робив (Рис. 3.1.3) та яка ідея цього сайту.

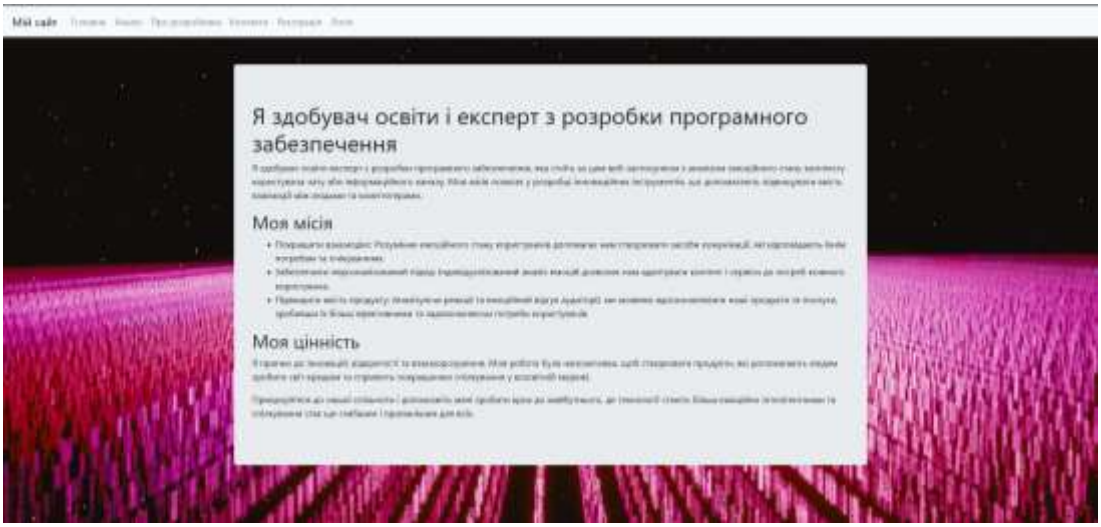


Рисунок 3.1.3 Сторінка про розробника сайту

Після цього можна побачити контакти з тим хто створював цей веб-застосунок (Рис. 3.1.4).

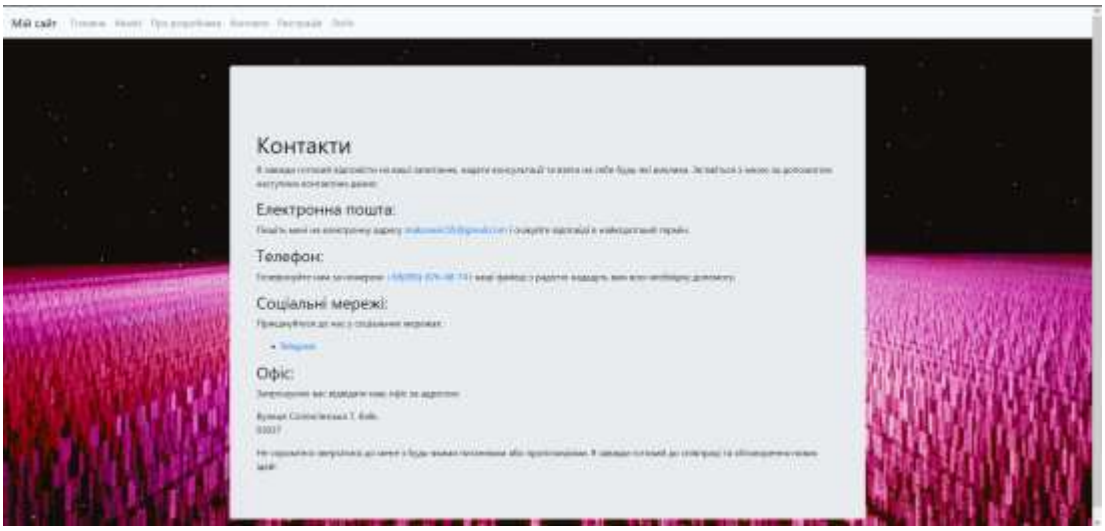


Рисунок 3.1.4 Сторінка контакти сайту

Перейдемо вже до головної ідеї веб-застосунку який створювався для аналізом емоційного стану контексту користувача чату або інформаційного каналу використовуючи мову програмування Python (Рис. 3.1.5) також на веб-застосунку реалізована система входу в акаунт зі своїми запитами (Рис. 3.1.6) та регістрація на сайті щоб можна було запам’ятовувати запити до аналізатора певного користувача (Рис. 3.1.7). Якщо користувач не авторизувався на сайт то його дані зберігатися не будуть.

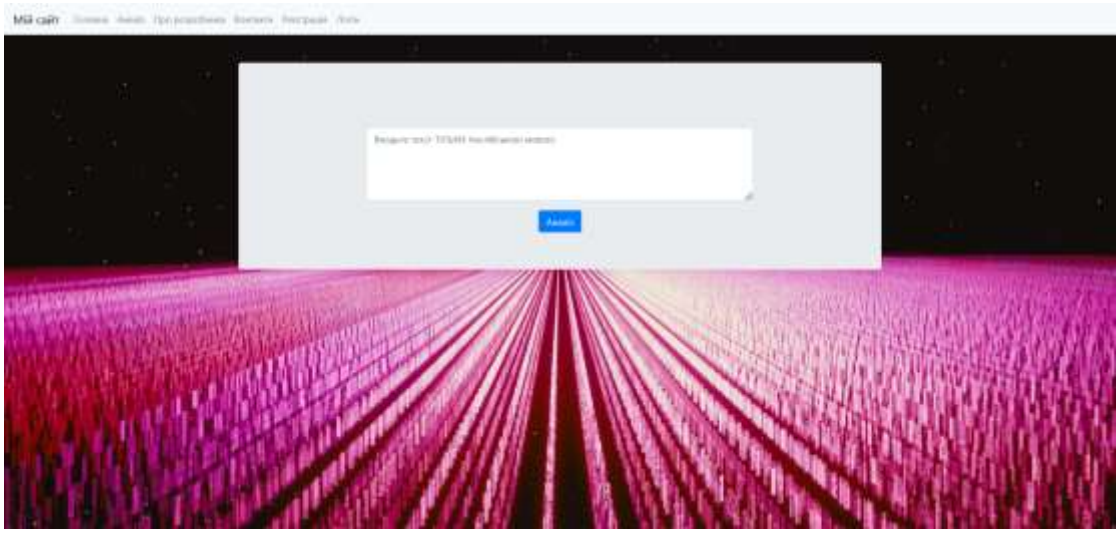


Рисунок 3.1.5 Сторінка з аналізом на сайті

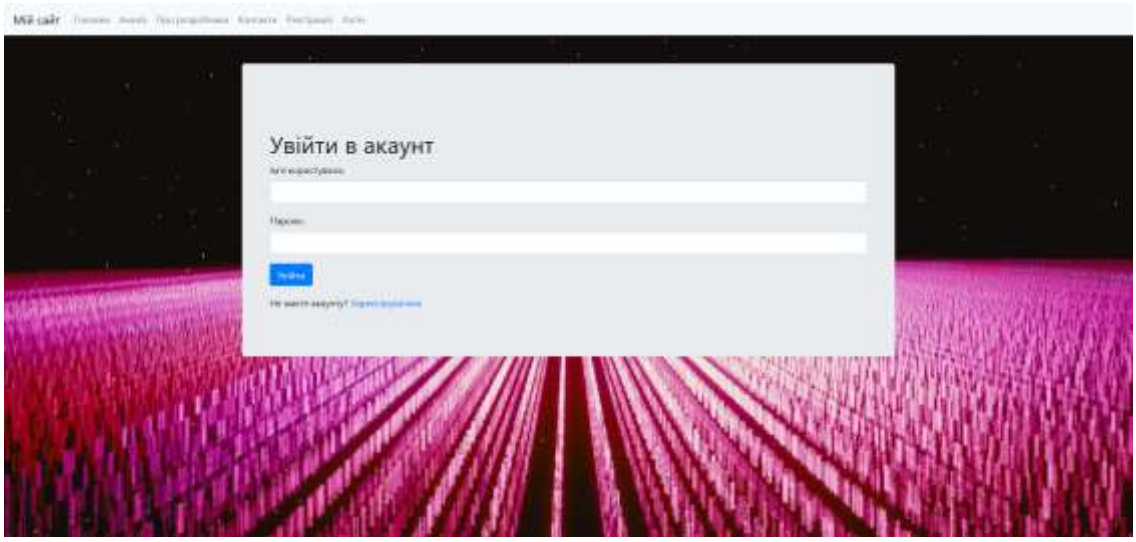


Рисунок 3.1.6 Сторінка з входом на сайт

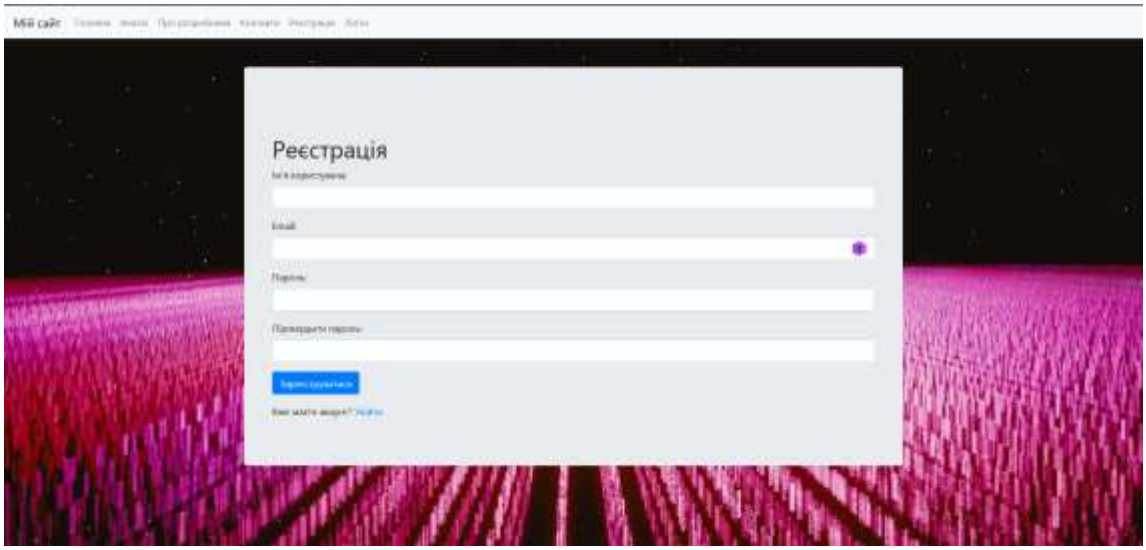


Рисунок 3.1.7 Сторінка з регістрацією на сайт

3.2 Код

Спочатку переглянемо головний шаблонний файл HTML (Рис. 3.2.1) та його компоненти які його наслідують та доповнюють по типу головної сторінки і тд. Для цього треба переглянути (Рис. 3.2.1 - 3.2.6)

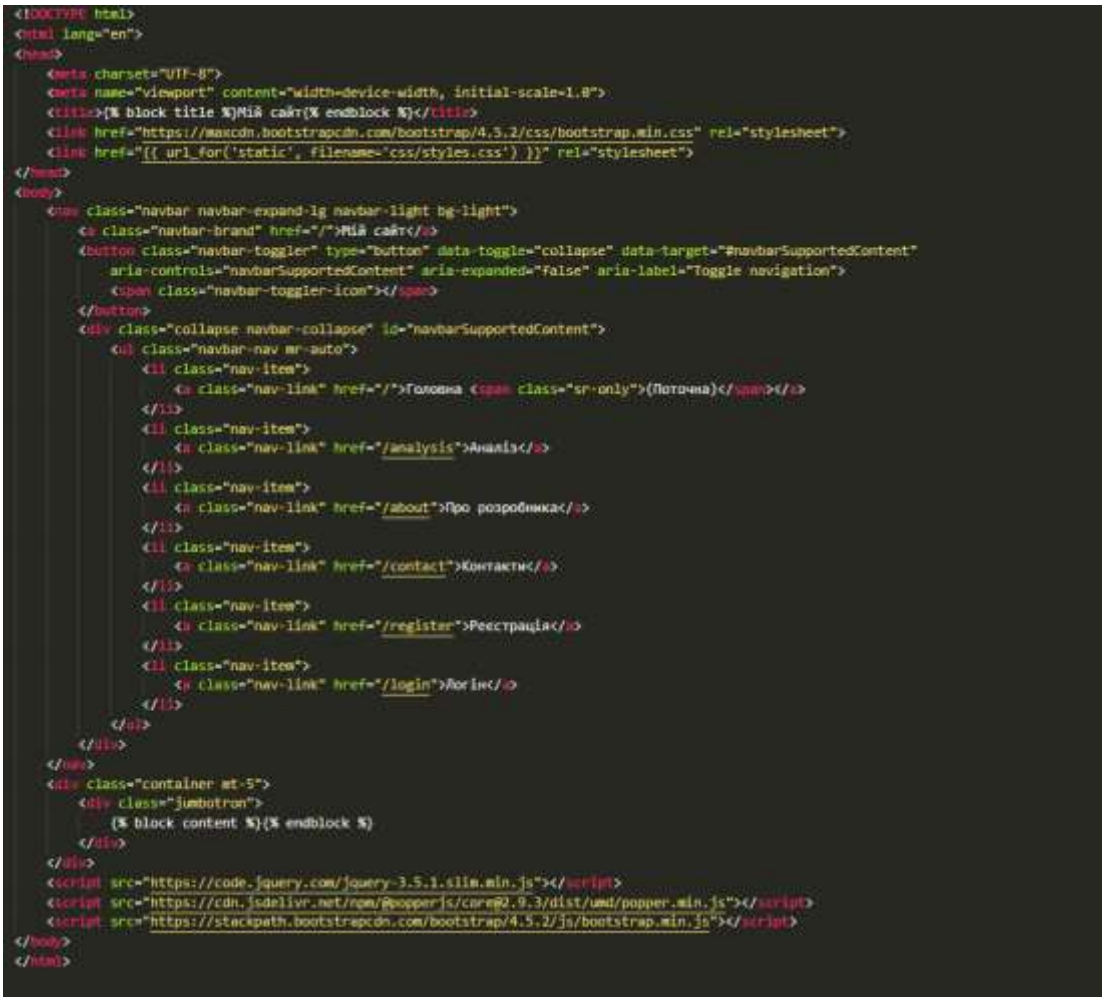


Рисунок 3.2.1 Рисунок коду фалу base.html

Рисунок 3.2.2 Рисунок коду фалу index.html

Рисунок 3.2.3 Рисунок коду фалу about.html

Рисунок 3.2.4 Рисунок коду фалу contact.html

```
{% extends 'base.html' %}

{% block title %}Увійти{% endblock %}

{% block content %}
<div class="container mt-5">
    <h1>Увійти в акаунт</h1>
    <form action="/login" method="post">
        <div class="form-group">
            <label for="username">Ім'я користувача:</label>
            <input type="text" class="form-control" id="username" name="username" required>
        </div>
        <div class="form-group">
            <label for="password">Пароль:</label>
            <input type="password" class="form-control" id="password" name="password" required>
        </div>
        <button type="submit" class="btn btn-primary">Увійти</button>
    </form>
    <p class="mt-3">Не маєте акаунту? <a href="/register">Зареєструватися</a></p>
</div>
{% endblock %}
```

Рисунок 3.2.5 Рисунок коду фалу login.html

```
{% extends 'base.html' %}

{% block title %}Реєстрація{% endblock %}

{% block content %}
<div class="container mt-5">
    <h1>Реєстрація</h1>
    <form action="/register" method="post">
        <div class="form-group">
            <label for="username">Ім'я користувача:</label>
            <input type="text" class="form-control" id="username" name="username" required>
        </div>
        <div class="form-group">
            <label for="email">Email:</label>
            <input type="email" class="form-control" id="email" name="email" required>
        </div>
        <div class="form-group">
            <label for="password">Пароль:</label>
            <input type="password" class="form-control" id="password" name="password" minlength="8" required>
        </div>
        <div class="form-group">
            <label for="confirm_password">Підтвердити пароль:</label>
            <input type="password" class="form-control" id="confirm_password" name="confirm_password" minlength="8" required>
        </div>
        <button type="submit" class="btn btn-primary">Зареєструватися</button>
    </form>
    <p class="mt-3">Вже маєте акаунт? <a href="/login">Увійти</a></p>
</div>
{% endblock %}
```

Рисунок 3.2.6 Рисунок коду фалу login.html

Тепер треба перейти до Python складової який буде обробляти запити та відповідати на низ залежно від запиту (Рис. 3.2.7 - 3.2.9)

```
from nltk.sentiment import SentimentIntensityAnalyzer
import json
import nltk
nltk.download('vader_lexicon')


def read_JSON_file(file_name):

    with open(file_name) as json_file:
        data = json.load(json_file)

        return data


def read_TXT_file(file_name):

    with open(file_name, "r", encoding='utf-8') as f:
        text = f.read().replace("\n\n", "").replace(
            "\n", "").replace("\t", "").replace("\t\t", "")

        return text


def sentiment_analiz(text):
    sia = SentimentIntensityAnalyzer()

    sentiment_scores = sia.polarity_scores(text)
    compound_score = sentiment_scores['compound']

    return compound_score


def emotion_in_text(file_json,file_txt):
    emotions = []

    for emotion, terms in file_json.items():
        if any(term in file_txt for term in terms):
            emotions.append(emotion)

    return(emotions)
```

Рисунок 3.2.7 Рисунок коду analizText.py

```python
from flask import Flask, render_template, request
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
from analizText import sentiment_analiz

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
db = SQLAlchemy(app)

class Users(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_name = db.Column(db.String(120), nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(60), nullable=False)
    date_created = db.Column(db.DateTime, default=datetime.utcnow)

    def __repr__(self):
        return f"Users('{self.user_name}', '{self.email}', '{self.date_created}')"


class Analysis(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
    text = db.Column(db.Text, nullable=False)
    analysis_result = db.Column(db.String(120), nullable=False)
    date_created = db.Column(db.DateTime, default=datetime.utcnow)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/home')
def home():
    return render_template('home.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/analysis', methods=['GET', 'POST'])
def analysis():
    if request.method == 'POST':
        text = request.form['myTextarea']
        analysis_result = sentiment_analiz(text)
        user_id = 1
        new_analysis = Analysis(user_id=user_id, text=text, analysis_result=analysis_result)
        db.session.add(new_analysis)
        db.session.commit()
        return render_template('analysis.html', text=text, analysis=analysis_result)
    return render_template('analysis.html')
```

Рисунок 3.2.8 Рисунок коду app.py Частина 1

```python
@app.route('/contact')
def contact():
    return render_template('contact.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        user_name = request.form['user_name']
        email = request.form['email']
        password = request.form['password']
        if Users.query.filter_by(email=email).first():
            return 'Email already registered'
        new_user = Users(user_name=user_name, email=email, password=password)
        db.session.add(new_user)
        db.session.commit()
        return 'User registered successfully'
    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        user = Users.query.filter_by(email=email).first()
        if user and user.password == password:
            return 'Login successful'
        else:
            return 'Login failed'
    return render_template('login.html')

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

Рисунок 3.2.9 Рисунок коду app.py Частина 2

ВИСНОВКИ

1. Досягнення мети

В ході виконання кваліфікаційної роботи було успішно розроблено веб-застосунок, що використовує мову програмування Python для аналізу емоційного стану контексту користувача чату або інформаційного каналу. Застосунок здатен визначати емоційне забарвлення тексту, класифікуючи його як негативне, нейтральне або позитивне.

2. Значення розробки

Ця розробка має значну цінність для людства, адже надає інструмент, який допомагає:

- Виявляти емоційну складову текстової комунікації в чатах та інформаційних каналах.
- Розуміти наміри та стан користувачів, що може бути корисно в багатьох сферах, наприклад, у сфері обслуговування клієнтів, освіти, онлайн-психології тощо.
- Відстежувати емоційний фон в чатах та на інформаційних ресурсах, що може допомогти у виявленні та попередженні конфліктів, мобінгу, поширення дезінформації тощо.

3. Подальший розвиток

Подальший розвиток веб-застосунку може йти за наступними напрямками:

- Розширення спектра емоцій, які може розпізнавати застосунок.
- Вдосконалення алгоритмів аналізу,
- Інтеграція з іншими системами,
- Розробка мобільного додатку.

4. Висновки

Розроблений веб-застосунок є перспективним інструментом для аналізу емоційного стану контексту користувача чату або інформаційного каналу. Його використання може суттєво покращити комунікацію та розуміння між людьми в онлайн-середовищі.

Важливо зазначити, що це лише приклад висновків. Ви можете доповнити його власними міркуваннями та висновками, які стосуються вашої конкретної кваліфікаційної роботи.

Ось декілька порад щодо доповнення висновків:

- Підкресліть практичну цінність вашого веб-застосунку.
- Опишіть потенційні сфери застосування вашого веб-застосунку.
- Вкажіть на можливі напрямки для подальшого розвитку вашого веб-застосунку.
- Сформулюйте чіткі та лаконічні висновки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Що таке емоції? Перелік емоцій (приклади) URL: https://dovidka.biz.ua/shho-take-emotsiyi-perelik-emotsiy-prikladi
2. Як керувати емоціями: поради психолога URL: https://hospodar.ua/post/4099-yak-keruvaty-emociyamy-porady-psyhologa-
3. Типи темпераменту людини URL: http://typesoftemp.blogspot.com/2015/03/blog-post.html
4. NLP vs NLU: What's The Difference? URL: https://www.bmc.com/blogs/nlu-vs-nlp-natural-language-understanding-processing/
5. HTML Tutorial URL: https://www.w3schools.com/html/
6. CSS Tutorial URL: https://www.w3schools.com/css/
7. JavaScript Tutorial URL: https://www.w3schools.com/js/
8. SQL Tutorial URL: https://www.w3schools.com/sql/
9. Python Tutorial URL: https://www.w3schools.com/python/
10. Bootstrap 4 Tutorial URL: https://www.w3schools.com/bootstrap4/
11. Емоційний стан людини та ненормативна лексика. Реферат URL: https://osvita.ua/vnz/reports/psychology/28082/
12. Гормони щастя та радості URL: https://esculab.com/hormony-shchastya-ta-radosti
13. Соцмережі забирають понад п'ять років нашого життя: нове дослідження URL: https://tsn.ua/svit/skilki-chasu-mi-provodimo-u-socialnih-merezhah-2361154.html
14. Епоха чатів: месенджери для інтернет-торгівлі URL: https://blog.keycrm.app/uk/epoha-chativ-mesendzheri-dlya-internet-torgivli/
15. Natural Language Assessment: A New Framework to Promote Educationhttps://research.google/blog/natural-language-assessment-a-new-framework-to-promote-education/

ДЕМОНСТАРЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА
на тему: «Розробка веб-застосунку з аналізом емоційного стану контексту користувача чату або інформаційного каналу використовуючи мову програмування PYTHON»

Виконав: здобувач вищої освіти гр. ШД-41
Максим ЯКУСЕВИЧ
Керівник: кандидат технічних наук, доцент
Максим ФЕСЕНКО

2024 рік

Мета роботи: Надати розроблений веб-застосунку для аналізування емоційного стану контексту користувача чату або інформаційного каналу використовуючи для розуміння емоційного стану спілкування або інформації

Об'єкт дослідження: Підвищити розуміння емоційного стану чату або інформаційного каналу

Предмет дослідження: Веб-застосунок для розуміння емоційного стану чату або інформаційного каналу.

ОСНОВНІ ЗАВДАННЯ РОБОТИ

1. Дослідити та провести аналіз емоційних станів людини та їх сприйняття.
2. Провести огляд існуючих технологій розроблення онлайн-консультантів, зокрема веб-технологій.
3. Обрати та обгрунтувати використання цих технологій для створення ефективного онлайн-консультанта.
4. Розробити веб-застосунок з аналізатором емоційних станів людини користувача чату або інформаційного каналу.

ЕМОЦІЇ ЛЮДИНИ

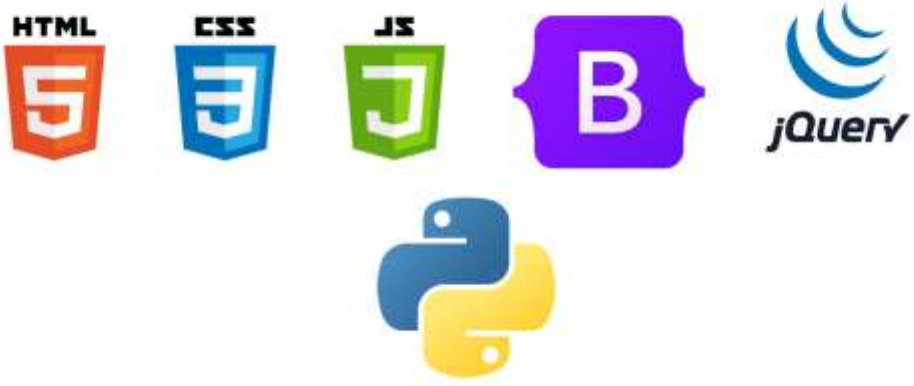
ДОСЛІДЖЕННЯ ФАКТОРІВ, ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБЛЕННЯ ВЕБ-ЗАСТОСУНКУ

- Комплекс почуттів
- Відношення до себе, до інших людей, до подій і предметів
- Позитивний, негативний або нейтральний
- Вплив на сприйняття, мислення, поведінку та взаємодію
- Темперамент людини

- Видалення зайвих символів
- Стемінг
- Лематизація
- Видалення стоп-слів

- Обробка природної мови (Natural language processing NLP)
- Розуміння природної мови (Natural language understanding NLU)
- Генерація природної мови (Natural language generation NLG)
- Оцінка природної мови (Natural Language Assessment NLA)

Вибір та обгрунтування технологій для сто



РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

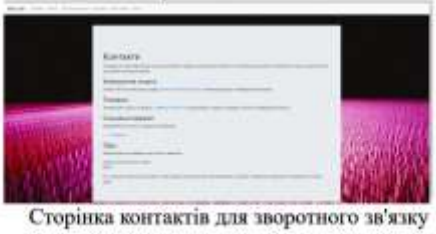
База даних запитів користувачі (One-To-One)

РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ

Головна сторінка сайту

Сторінка з аналізом

Сторінка про розробника сайту



Сторінка контактів для зворотного зв'язку

РЕАЛІЗАЦІЯ РЕЄСТРАЦІЇ З ВХОДОМ У СИСТЕМУ

Сторінка з реєстрацією до сайту

Сторінка з входом до сайту

ВИСНОВКИ

1. Досліджено та проведено аналіз емоційних станів людини та їх сприйняття.
2. Проведено огляд існуючих технологій для розроблення веб-застосунку з аналізом емоційного стану користувача.
3. Обрано технології для створення ефективного веб-застосунку для аналізу емоційного стану чату або інформаційного каналу .
4. Розроблено веб-застосунку з аналізатором емоційних станів людини користувача чату або інформаційного каналу.

Дякую за увагу!!!