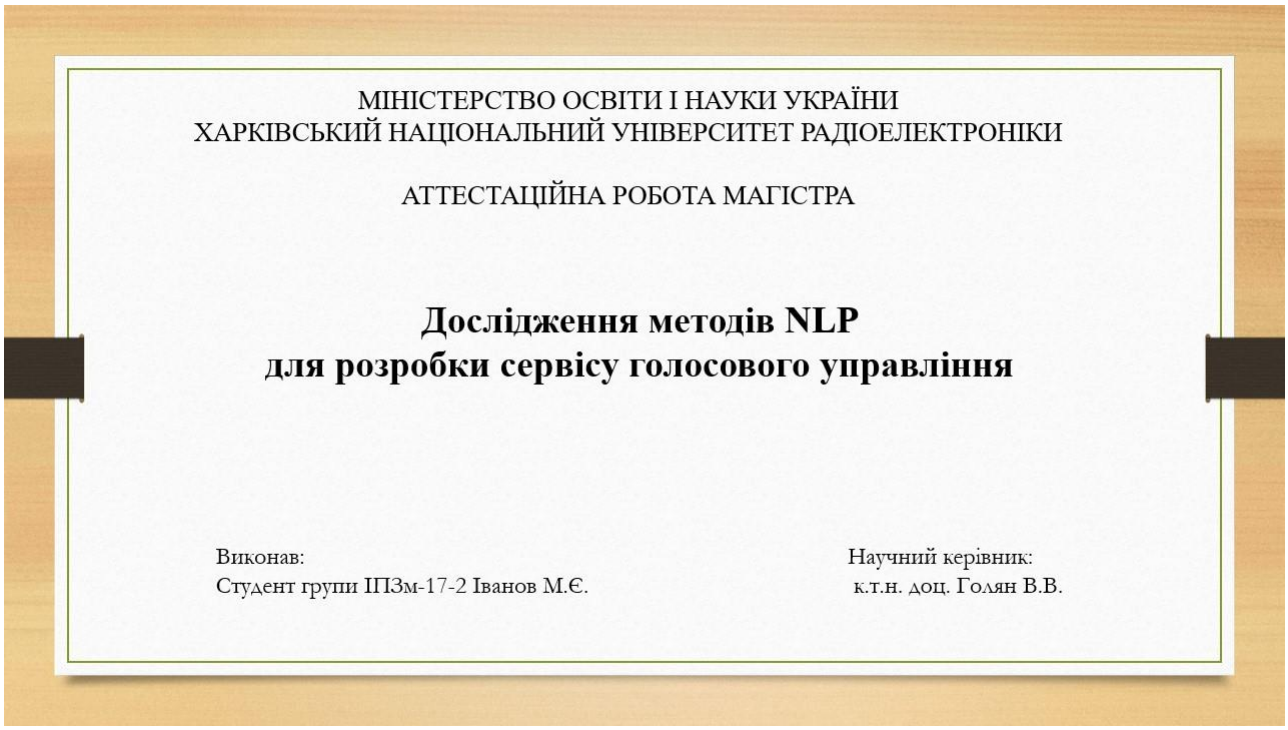
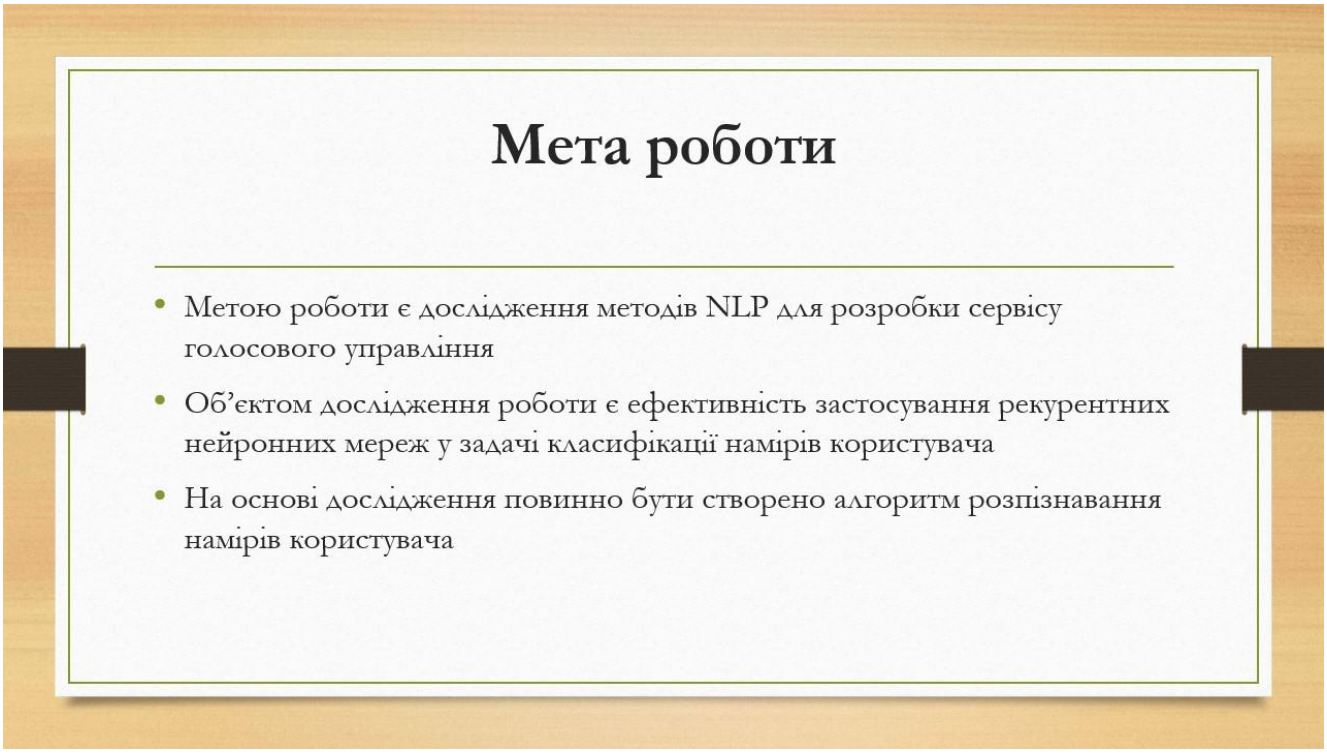
Додаток А

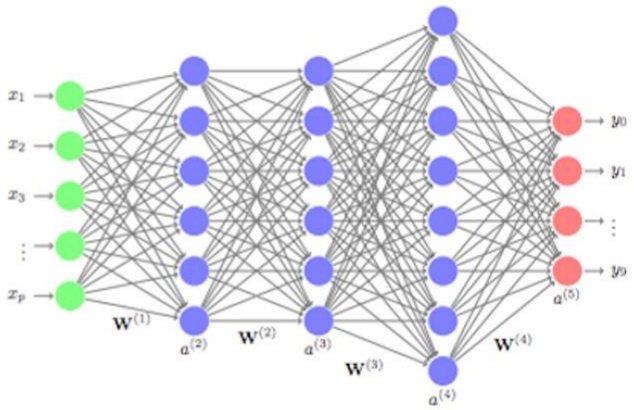
Слайди презентацій



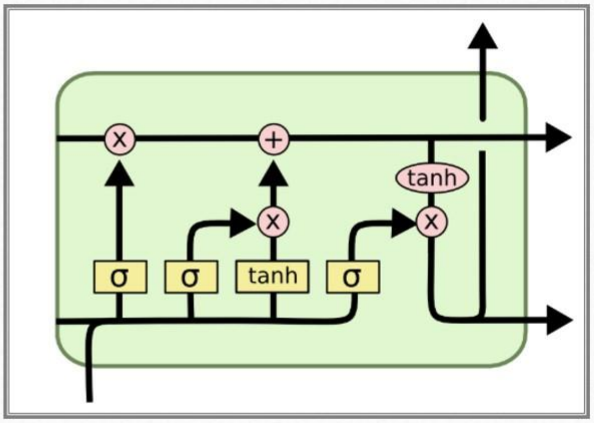


Використання нейронних мереж у якості методів NLP

Глибинні нейронні мережі



Мережа LSTM



Технології дослідження нейронних мереж

colab

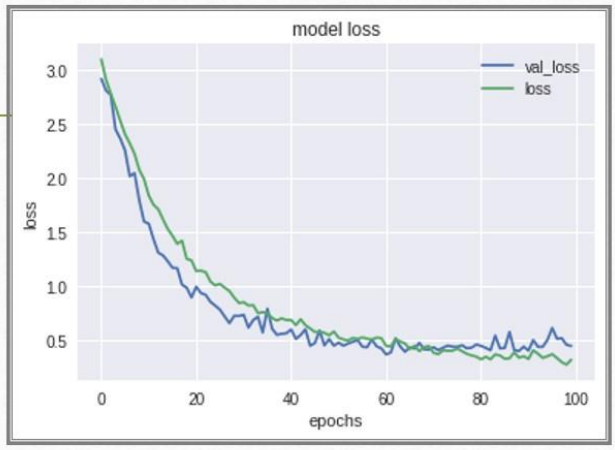
python™

Keras

- Забезпечення комплекту розробки, що дозволяє швидко створювати функціональні модулі
- Розширюваний фреймворк для комбінування та конфігурування загальних розробок
- Хмарне середовище, яке забезпечує виконання різноманітних обчислень

Дослідження ефективності LSTM для розпізнавання намірів

- 10000 одиниць у наборі даних з різноманітними намірами користувачів
- 80% даних для навчання, 20% для тестування
- 100 епох виконання
- Результати: 88% точності на тренувальному наборі та 86% на тестовому

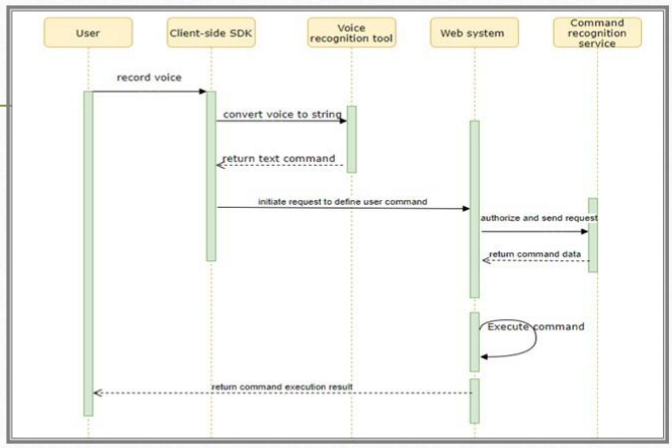


Алгоритм розпізнавання намірів користувача

Розробка сервісу голосового управління

Проектування сервісу голосового управління

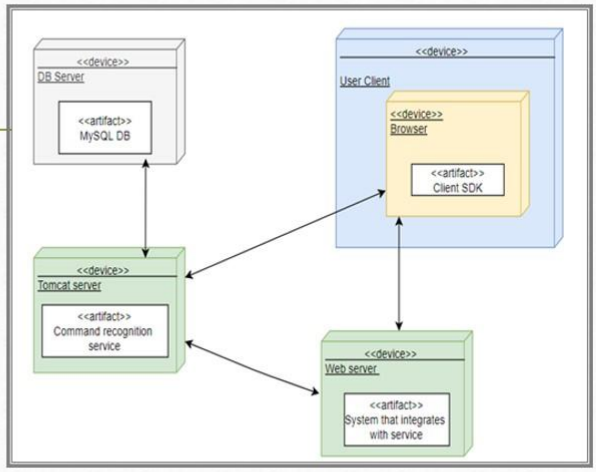
- Розділення розпізнавання голосу від обробки природної мови
- Інкапсуляція сервісу розпізнавання намірів користувача для спрощення контракту взаємодії
- Вбудована інтеграція між сервісом та системою, що його використовує



Вибір засобів розробки

Огляд розгортки сервісу голосового управління

- Артефакти: клієнтський SDK, сервіс розпізнавання команд та база даних
- Взаємодія через REST HTTP API
- Можливість виконання різних сценаріїв інтеграції

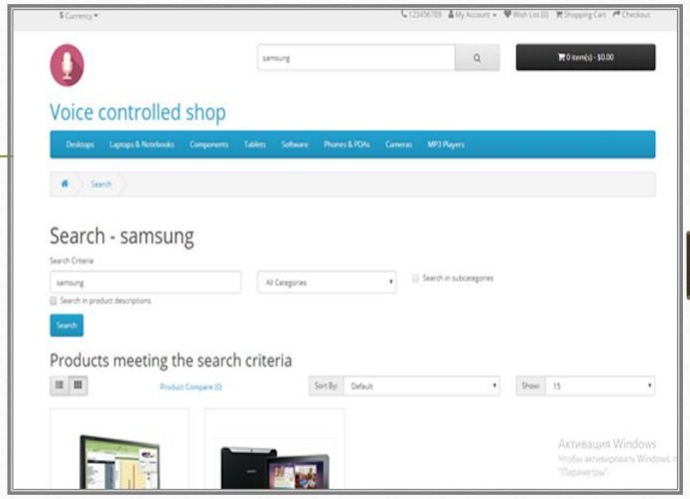


Огляд функцій сервісу голосового управління

Можливість здійснювати голосовий пошук

Можливість голосового заповнення форм

Можливість навігації за сторінками системи через голосові команди



Висновки

- Досліджено ефективність методів NLP на основі рекурентних нейронних мереж
- Розроблено комплексний алгоритм розпізнавання намірів користувача на основі класифікації за допомогою рекурентної нейронної мережі, модулю розпізнавання сутностей та модулю конфігурації шаблонів користувача
- Розроблено сервіс голосового управління, що використовує алгоритм розпізнавання намірів та сучасні засоби розробки і проектування

Додаток Б

Лістинг програмного коду

```
package ua.nure.diploma.controllers;

import org.springframework.web.bind.annotation.*;
import ua.nure.diploma.commands.Command;
import ua.nure.diploma.commands.CommandService;
import ua.nure.diploma.entities.Customer;
import ua.nure.diploma.repositories.IntentUserRepository;

import javax.annotation.Resource;

@RestController
@RequestMapping("/intents")
public class IntentManagementController {

    @Resource
    private IntentUserRepository intentUserRepository;
    @Resource
    private CommandService commandService;

    @GetMapping("/getcommand/{user}/{query}")
    public Command getCommand(@PathVariable String user, @PathVariable String query) {
        return commandService.getCommand(user, query);
    }

    @GetMapping("/intentusers")
    public Iterable<Customer> intentUsers() {
```

```
        return intentUserRepository.findAll();
    }

    @GetMapping("/createintentuser")
    public void createIntentUser(@RequestParam String name, @RequestParam String lastName) {
        Customer customer = new Customer(name, lastName, 1000);
        intentUserRepository.save(customer);
    }

    @GetMapping("/intentusers/{id}/[tokens}")
    public void setTokens(@PathVariable String id, @PathVariable int tokens) {
        Customer user = intentUserRepository.findByFirstName(id);
        user.setTokens(tokens);
        intentUserRepository.save(user);
    }

}
package ua.nure.diploma.entities;

import javax.persistence.*;

@Entity
public class Customer {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String firstName;
    private String lastName;
```

```
public int getTokens() {
    if(tokens == null){
        return 0;
    }
    return tokens;
}

public void setTokens(Integer tokens) {
    this.tokens = tokens;
}
@Column(nullable = true)
private Integer tokens = 10000;

protected Customer() {
}

public Customer(String firstName, String lastName, int tokens) {
    this.firstName = firstName;
    this.lastName = lastName;
}

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}
```

```
    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    @Override
    public String toString() {
        return String.format(
                "Customer[id=%d, firstName='%s', lastName='%s']",
                id, firstName, lastName);
    }

}
@SpringBootApplication(scanBasePackages = {"ua.nure"})
public class DiplomaApplication {
    private static final Logger log =
LoggerFactory.getLogger(DiplomaApplication.class);

    public static void main(String[] args) {
```

```
        SpringApplication.run(DiplomaApplication.class, args);
    }
def obtain_intent_dataset(path):

  df = pd.read_csv(path, encoding = "latin1", names = ["sent", "int"])
  print(df.head())
  intent = df["int"]
  unique_intent = list(set(intent))
  sentences = list(df["sent"])

  return (intent, unique_intent, sentences)
def init_m(vc_size, ml):
  model = Sequential()
  model.add(Embedding(vc_size, size, input_length = ml, trainable = False))
  model.add(Bidirectional(LSTM(size)))
  model.add(Dense(dns_size, activation = "relu"))
  model.add(Dropout(half_dropout))
  model.add(Dense(21, activation = "softmax"))

  return model
filename = ‘resultingmd.h5'
checkpoint = ModelCheckpoint(path, monitor='val_loss', verbose=1,
save_best_only=True, mode='min')

hist = model.fit(XTRN,YTRN, epochs = cfg_epchs, batch_size = cfg_btch,
validation_data = (XVAL, YVAL), callbacks = [checkpoint])
```