Hafta 2

Chat Bota Cevap Verebileceği Soru ekleme

1. nlu.yml dosyası:

```
nlu:

- intent: ask_best_proglang
  examples: |
    - What is the best programming language?
    - which programming language is the best?
    - What is the top programming language?
```

bu dosyaya nlu başlığı altında kullanıcı ile girilebilecek bir diyalog senaryosunun binevi ismi yazılır. Şu şekilde örnekler olabilir: greet, goodbye, affirm, deny vs.

2. rules.yml dosyası:

```
rules:

- rule:
  steps:
  - intent: ask_best_proglang
  - action: utter_best_language
```

bu dosyaya rules başlığı altında belli başlı rule'lar yazılır. Yine burada ekleyeceğimiz yeni diyalogun prompt-response (mesaj geldiğinde alınacak aksiyon) sırası için bir steps dizisi tanımlanır. Genel olarak utter_ öneki eylem gerektiren yerlerde isim için kullanılır.

3. stories.yml dosyası:

```
stories:

- story: respond the best language question
  steps:
  - intent: ask_best_proglang
  - action: utter_best_language
```

bu dosyada da hikayenin ismi "respond the best language question" olarak belirlenir ve sonrasında gelen prompta karşılık yine steps dizisi sıralanır.

4. domain.yml dosyası:

```
intents:

  - ask_best_proglang

responses:

  utter_best_language:
  - text: "Absolutely C"
```

bu dosya da botun genel yapılandırmasını tanımlar ve niyetler, yanıtlar ve eylemler gibi bileşenleri içerir.

Şimdi bunları kaydettikten sonra yeni haliyle botu derliyoruz.

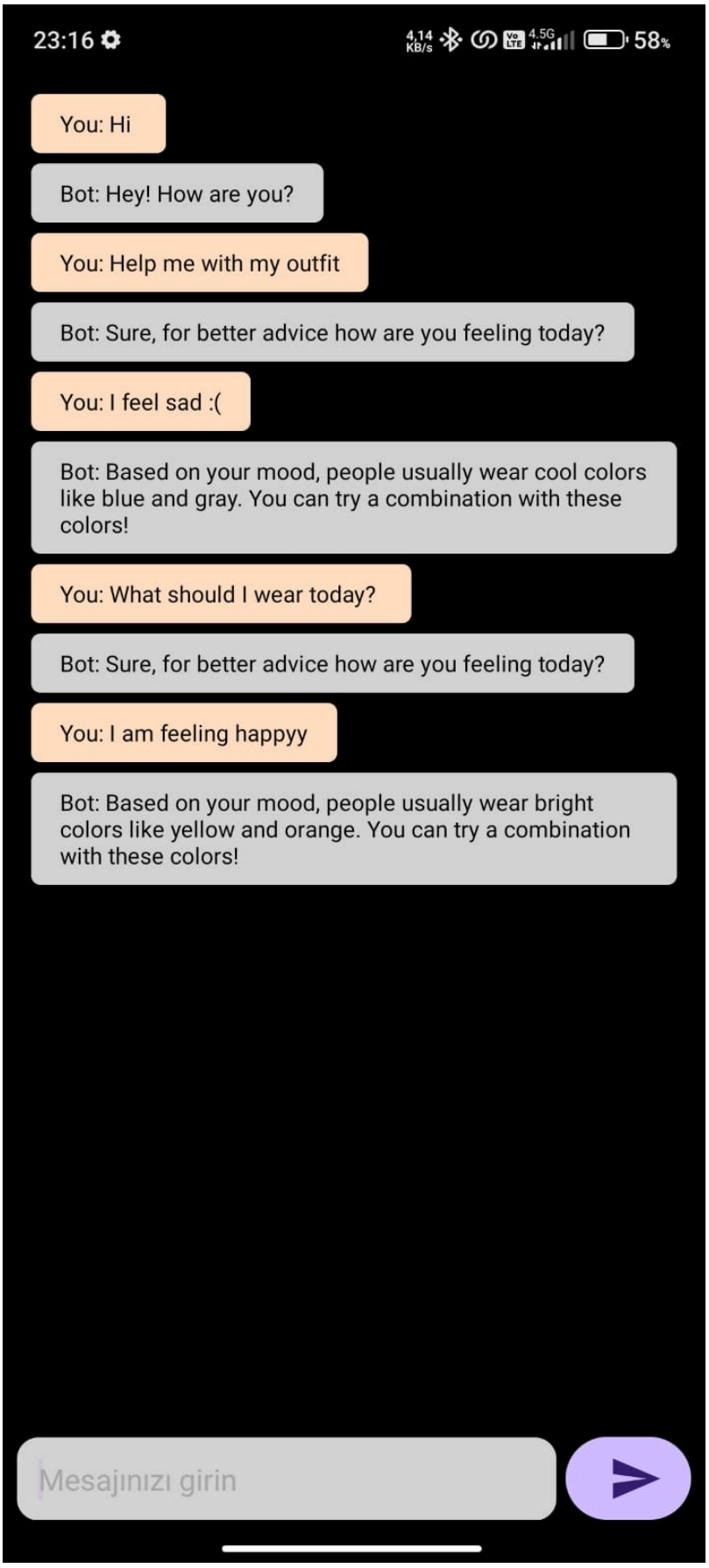
```
rasa train
rasa run actions
rasa shell
```

burada dikkat edilmesi gereken şey şudur: "rasa run actions" komutundan sonra gelecek "rasa shell" komutu ayrı bir terminal penceresinden girilmelidir.

Chat Bot ile Dinamik Sohbet Oluşturma

Bugün üzerinde çalıştığım bota, verilen prompta göre bir soru sorup, bu soruya verdiğim cevaba göre dinamik bir yanıt vermesini sağlayacak kodlar ekleyeceğim.

Bota bugünkü kombinime yardım etmesini söyleyeceğim bot da karşılık olarak o anki ruh halimi soracak. Verdiğim cevaba göre ruh halime göre hangi renk giyebileceğimi önerecek.



Şimdi bunları eklemek için projede yaptığımız değişiklikleri ardından bu değişiklikleri yaparken karşılaştığımız hataları sıralayacağım.

1. nlu.yml dosyasında değişiklikler:

Aşağıdaki satırları ekliyoruz.

```
- intent: ask_clothing_advice
  examples: |
    - Help me with my outfit
    - What should I wear today?
    - Can you give me clothing advice?
    - I need help with my clothing
    - What should I wear based on my mood?
```

2. stories.yml dosyasında değişiklikler:

Aşağıdaki satırları ekliyoruz.

```
- story: Provide clothing advice
  steps:
  - intent: ask_clothing_advice
  - action: utter_ask_mood
  - intent: inform_mood
  - action: action_clothing_advice
```

3. rules.yml dosyasında değişiklikler:

Aşağıdaki satırları ekliyoruz.

```
- rule: Ask for mood before giving clothing advice
  steps:
  - intent: ask_clothing_advice
  - action: utter_ask_mood
```

4. domain.yml dosyasında değişiklikler:

a) intents başlığı altına aşağıdaki satırları ekliyoruz:

```
  - ask_clothing_advice
  - inform_mood
```

b) responses başlığı altına aşağıdaki satırları ekliyoruz:

```
  utter_ask_mood:
  - text: "Sure, for better advice how are you feeling today?"

  utter_clothing_advice:
  - text: "Based on your mood, people usually wear {colors}. You can try a
combination with these colors!"
```

c) Daha önce var olmayan slots yeni bir bölüm ekliyoruz

```
slots:
  mood:
    type: text
    influence_conversation: true
    mappings:
    - type: from_text
```

slots yapısı sohbet süreci içinde kullanıcıdan gelen promptlarda gerekli olan değişkenleri tutar. Burada tutacağımız değişken kullanıcının modu olacak. O yüzden mood adında bir isim verip bunun türünü belirliyoruz. Ayrıca influence_conversation değerinin true olması bu soruya cevap verilmesi gerektiğini belirtir. Aksi takdirde herhangi bir cevap verilmez bot tarafından.

d) Actions kısmını eklemek

Bu yeni eklenen özellik özel bir eylem gerektirdiği için (kullanıcıdan gelen cevaba göre hareket etmek) domain.yml dosyasında diğer komutlar için var olmayan actions başlığı da eklenir. domain.yml dosyasında actions kısmına özel eylemleri (custom actions) eklemek, Rasa'nın bu eylemleri tanımasını ve çağırabilmesini sağlar. Eğer domain.yml dosyasına bu eylemi eklemezsek, Rasa bu özel eylemin varlığını bilemez ve çağırdığında hata alırız.

```
actions:
  - action_clothing_advice
```

Diğer sohbetler için neden böyle bir şey yok?

- Basit yanıtlar ve diyaloglar sadece responses kısmında tanımlanan hazır cevaplar kullanılarak işlenir. Bu yanıtlar için ekstra bir Python kodu gerekmez.
- Özel bir eylem yazmanız gerektiğinde (örneğin, kullanıcıya özel bir tavsiye vermek gibi), bu eylemin çalıştırılabilmesi için domain dosyasına eklenmesi gerekir.

Özel Eylemler ve Hazır Yanıtlar Arasındaki Fark

- Hazır Yanıtlar (Utterances): Basit metin cevapları ve önceden tanımlanmış mesajları içerir. utter_ prefix ile kullanılır ve domain dosyasındaki responses kısmında tanımlanır.
- Özel Eylemler (Actions): Dinamik olarak hesaplanan veya özel iş mantığına sahip yanıtlar. Bu eylemler Python kodunda tanımlanır ve domain dosyasındaki actions kısmına eklenir.

Özel Eylem Gerektirebilecek Durumlar:

- Kullanıcının verisine dayalı hesaplamalar yapma.
- Üçüncü parti API'leri çağırma.

- Karmaşık iş mantığı içeren cevaplar oluşturma.

5. endpoints.yml dosyasındaki değişiklikler:

endpoints.yml dosyasına özel eylem sunucusunun (action server) URL'sini eklemek, Rasa'nın bu özel eylemleri çalıştırabilmesi için gereklidir. Özel eylemler, Rasa'nın standart yanıtları dışında belirli bir iş mantığını çalıştıran Python kodlarıdır ve bunların çalıştırılabilmesi için bir sunucuya ihtiyaç vardır. İşte bu nedenle, endpoints.yml dosyasına aşağıdaki gibi bir giriş eklenir:

```
action_endpoint:
  url: "http://localhost:5055/webhook"
```

- (Burada localde çalışıldığı için kafama göre bir port numarası verdim.)

endpoints.yml dosyasına özel eylem sunucusunun URL'sini eklemek, Rasa'nın bu özel eylemleri çalıştırabilmesini sağlar. Özel eylemler, standart yanıtların ötesinde belirli iş mantıklarını çalıştıran Python kodlarıdır ve bunların yürütülmesi için bir sunucuya ihtiyaç vardır. Bu nedenle, endpoints.yml dosyasına action server'ın URL'si eklenir.

Bu ekleme sayesinde, Rasa botu bir özel eylemi çağırdığında, action server'a doğru yönlendirilir ve istenen iş mantığı çalıştırılır.

Kısaca URL adresi için açıklama:

http://localhost:5055/webhook URL'si, Rasa'nın özel eylemlerini (custom actions) çalıştıracak olan eylem sunucusunun adresidir. Bu URL, Rasa'nın özel eylemleri çalıştırmak için iletişim kuracağı sunucuyu belirtir. İşte detaylı bir açıklama:

Özel Eylem Sunucusu (Action Server)

Adres: http://localhost:5055/webhook
Port: 5055
Yol (Path): /webhook

Ne İşe Yarar?

- Özel Eylem Sunucusunun Çalıştırılması:
- Özel eylemler, Python kodları ile yazılır ve belirli iş mantıklarını içerir.
- Bu özel eylemler, Rasa botunun standart yanıtlarının ötesinde daha karmaşık işlemler yapmasını sağlar.
- Özel eylemler, actions.py dosyasında tanımlanır.
- Eylem Sunucusunun Çalıştırılması:
- rasa run actions --port 5055 komutuyla özel eylem sunucusunu çalıştırırsınız.
- Bu sunucu, Rasa'nın özel eylem isteklerini alır ve işleme koyar.
- Rasa'nın Eylem Sunucusuna Erişimi:
- endpoints.yml dosyasındaki action_endpoint girişi, Rasa'ya özel eylem sunucusunun nerede çalıştığını söyler.
- Örneğin, http://localhost:5055/webhook URL'si, Rasa'nın özel eylem isteklerini bu adrese göndermesini sağlar.
- localhost ifadesi, sunucunun yerel bilgisayarınızda çalıştığını belirtir.

- 5055 port numarası, bu sunucunun hangi port üzerinden erişilebilir olduğunu belirtir.
- /webhook yolu, isteklerin bu sunucunun hangi uç noktasına (endpoint) gönderileceğini belirtir.

6. actionsy.py dosyasında değişiklikler:

```
from typing import Any, Text, Dict, List
from rasa_sdk import Action, Tracker
from rasa_sdk.executor import CollectingDispatcher
from rasa_sdk.events import SlotSet

class ActionClothingAdvice(Action):

    def name(self) -> Text:
        return "action_clothing_advice"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

        mood = tracker.get_slot('mood')

        # Debugging: Print the value of the 'mood' slot
        print(f"DEBUG: Mood slot value is: {mood}")

        # Basit bir ruh hali ve renk eşlemesi
        mood_to_colors = {
            "happy": "bright colors like yellow and orange",
            "sad": "cool colors like blue and gray",
            "excited": "vibrant colors like red and purple",
            "nervous": "calm colors like green and beige",
            "calm": "neutral colors like white and light blue"
        }

        if mood:
            mood_cleaned = mood.lower()
            if "happy" in mood_cleaned:
                colors = mood_to_colors["happy"]
            elif "sad" in mood_cleaned:
                colors = mood_to_colors["sad"]
            elif "excited" in mood_cleaned:
                colors = mood_to_colors["excited"]
            elif "nervous" in mood_cleaned:
                colors = mood_to_colors["nervous"]
            elif "calm" in mood_cleaned:
                colors = mood_to_colors["calm"]
            else:
                colors = "a mix of colors"
        else:
            colors = "a mix of colors"

        dispatcher.utter_message(text=f"Based on your mood, people usually wear
{colors}. You can try a combination with these colors!")
```

```
        return [SlotSet("mood", mood)]
```

actions.py dosyasında, botun standart yanıtlarının ötesinde belirli iş mantıklarını gerçekleştirmek için özel eylemler (custom actions) tanımlandı. Bu dosyada, kullanıcıdan gelen verilere dayalı olarak dinamik yanıtlar oluşturabilen bir özel eylem tanımlanmışır.

- Slot Verisinin Alınması:
- mood = tracker.get_slot('mood') satırı ile kullanıcının ruh hali mood slotundan alınır.
- Debugging amacıyla mood slotunun değeri yazdırılır (print(f"DEBUG: Mood slot value is: {mood}")).
- Ruh Hali ve Renk Eşlemesi:
- mood_to_colors sözlüğü, çeşitli ruh hallerine karşılık gelen renkleri tanımlar.
- mood_cleaned değişkeni ile mood değeri küçük harflere dönüştürülür ve eşleşme yapılabilir forma getirilir.
- if-elif blokları ile mood_cleaned içindeki anahtar kelimelere göre uygun renkler seçilir. Anahtar kelime eşleştirme sırasında, kullanıcının girdiği metin içinde anahtar kelimeler aranır (örneğin, "happy", "sad").

Karşılaşılan Sıkıntılar ve Çözümler

1. endpoitns.yml Yapılandırması Eksikliği:

Başlangıçta, özel eylem sunucusunun URL'sini endpoints.yml dosyasına eklememiştim.

action_endpoint:
  url: "http://localhost:5055/webhook"

bu satırı ekleyerek düzelttim.

2. Slot Değerinin Alınamaması:

Kullanıcının ruh halinin alınamaması ve slot değerinin boş kalması problemi yaşandı.
Çözüm: Slot değerinin alındığını doğrulamak için debugging kodları ekledim ve mappings kısmında from_text ile slot değeri alınmasını sağladım.

3. Port Çakışması Problemi:

Birden fazla terminalde aynı port kullanımı nedeniyle hata alındı.
Çözüm: Farklı terminaller için farklı port numaraları kullanarak çalıştırdım.

Terminal 1: rasa run actions --port 5055
Terminal 2: rasa run --endpoints endpoints.yml --port 5005
Terminal 3: rasa shell --port 5006

Eklenen yeni özellikler sonrasında yeni modelin çalıştırılması

3 ayrı terminalde işlem yapılmalı:

```
# terminal 1
rasa train
rasa run actions --port 5055

# terminal 2
rasa run --endpoints endpoints.yml --port 5005

# terminal 3
# for linux terminal
rasa shell --port 5006
# for android program
rasa run --enable-api --cors "" --port 5006
```