Yapay Zeka Tabanlı Web Uygulaması Geliştirme ve Dağıtım

Tahmini Süre: 60 dakika

Genel Bakış

Bu projede, sağlanan bir metin üzerinde duygu analizi yapacak bir uygulama oluşturmak için gömülü Watson AI kütüphanelerini kullanıyoruz. Ardından, bu uygulamayı Flask çerçevesini kullanarak web üzerinde dağıtıyoruz.

Proje yönergeleri

Bu projeyi tamamlamak için, kurs boyunca edindiğiniz bilgiye dayalı olarak aşağıdaki 8 görevi tamamlamanız gerekecek.

Görevler ve hedefler:

- Görev 1: Proje deposunu klonlayın
- Görev 2: Watson NLP kütüphanesini kullanarak bir duygu analizi uygulaması oluşturun
- Görev 3: Uygulamanın çıktısını formatlayın
- Görev 4: Uygulamayı paketleyin
- Görev 5: Uygulamanızda birim testleri çalıştırın
- Görev 6: Flask kullanarak web uygulaması olarak dağıtın
- Görev 7: Hata yönetimini dahil edin
- Görev 8: Statik kod analizi yapın

Hadi başlayalım!

Gömülebilir Watson AI Kütüphaneleri Hakkında

Bu projede, AI destekli bir Python uygulaması oluşturmak için gömülebilir kütüphaneleri kullanacaksınız.

Gömülebilir Watson AI kütüphaneleri NLP kütüphanesini, metinden sese kütüphanesini ve sesten metne kütüphanesini içerir. Bu kütüphaneler, uygulamanızın bir parçası olarak gömülebilir ve dağıtılabilir. Kolaylık sağlaması açısından, bu kütüphaneler bu proje için Skills Network Labs Cloud IDE üzerinde önceden yüklenmiştir.

NLP kütüphanesi, duygu analizi, duygu tespiti, metin sınıflandırması, dil tespiti gibi işlevleri içerir. Sesten metne kütüphanesi, transkripsiyon hizmetini gerçekleştiren ve konuşulan seslerden yazılı metin üreten işlevleri içerir. Metinden sese kütüphanesi, yazılı metinden doğal sesli sesler üretir. Bu kütüphanelerin her birindeki mevcut işlevler, Cloud IDE sunucularında bulunan ve tüm kullanıcılar için ücretsiz olarak erişilebilen önceden eğitilmiş AI modellerini çağırır.

Bu kütüphanelere kişisel sistemleriniz üzerinden de erişilebilir. Bununla ilgili yönergeler Watson AI kütüphane sayfasında mevcuttur.

Görev 1: Proje deposunu klonlayın

Projenin Github deposu aşağıda belirtilen URL'de mevcuttur.

```
https://github.com/ibm-developer-skills-network/zzrjt-practice-project-emb-ai.git
```

- Yeni bir Terminal açın ve `mkdir` komutunu kullanarak `practice_project` dizinini oluşturun ve `cd` komutunu kullanarak mevcut dizini `practice_project` olarak değiştirin.

```
mkdir practice_project
cd practice_project
```

- Bu GitHub reposunu, Cloud IDE terminalini kullanarak projenize `practice_project` adında bir klasöre klonlayın.

▶ İpucu için buraya tıklayın
▶ Çözüm için buraya tıklayın

- Python 3.11 ve gerekli kütüphanelerin mevcut olduğundan emin olun:

```
python3.11 -V
pip3.11 show requests flask pylint
```

- Eksik kütüphaneleri yükleyin:

```
python3.11 -m pip install requests flask pylint
```

- Tamamlandığında, proje sekmesinde klasör yapısının resimde gösterildiği gibi olması gerekir.

Görev 2: Watson NLP kütüphanesini kullanarak bir duygu analizi uygulaması oluşturma

NLP duygu analizi, bir metinde ifade edilen duygu veya hisleri tanımak için bilgisayarların kullanılmasını içeren bir uygulamadır. NLP aracılığıyla, duygu analizi kelimeleri olumlu, olumsuz veya nötr olarak kategorize eder.

Duygu analizi genellikle metin verileri üzerinde gerçekleştirilir, böylece işletmeler müşteri geri bildirimlerinde marka ve ürün duygu durumunu izleyebilir ve müşteri ihtiyaçlarını anlayabilir. Bu, daha geniş bir kamuoyunun tutumunu ve ruh halini anlamaya yardımcı olur ve böylece bağlam hakkında içgörüler toplamak için yararlı bilgiler sağlar.

Duygu analizi uygulamasını oluşturmak için Watson Embedded AI Kütüphanelerini kullanacağız. Bu kütüphanelerin işlevleri zaten Cloud IDE sunucusunda dağıtıldığından, bu kütüphaneleri kodumuza dahil etmemize gerek yoktur. Bunun yerine, gerekli metinle birlikte ilgili modele bir POST isteği göndermemiz gerekiyor ve model uygun yanıtı gönderecektir.

Böyle bir uygulama için örnek bir kod şu şekilde olabilir

```
import requests
def <function_name>(<input_args>):
    url = '<relevant_url>'
    headers = {<header_dictionary>}
    myobj = {<input_dictionary_to_the_function>}
    response = requests.post(url, json = myobj, headers=header)
    return response.text
```

Not: Watson NLP fonksiyonlarının yanıtı bir nesne şeklindedir. Yanıtın detaylarına erişmek için, nesnenin `text` niteliğini kullanarak `response.text` çağrısı yapabilir ve fonksiyonun yanıtı basit metin olarak döndürmesini sağlayabiliriz.

Bu proje için, Watson NLP Kütüphanesi'nin BERT tabanlı Duygu Analizi fonksiyonunu kullanacaksınız. Bu fonksiyona erişmek için URL, başlıklar ve girdi json formatı aşağıdaki gibidir.

```
URL: 'https://sn-watson-sentiment-bert.labs.skills.network/v1/watson.runtime.nlp.v1/NlpService/SentimentPredict'
Headers: {"grpc-metadata-mm-model-id": "sentiment_aggregated-bert-workflow_lang_multi_stock"}
Input json: { "raw_document": { "text": text_to_analyse } }
```

Burada, `text_to_analyze` analiz edilecek gerçek yazılı metni tutan bir değişken olarak kullanılmaktadır.

Bu görevde, `practice_project` klasöründe `sentiment_analysis.py` adında yeni bir dosya oluşturmanız gerekiyor. Bu dosyada, yukarıda tartışıldığı gibi Watson NLP BERT Duygu Analizi fonksiyonunu kullanarak duygu analizi gerçekleştiren bir fonksiyon yazın. Bu fonksiyonu `sentiment_analyzer` olarak adlandıralım. Analiz edilecek metnin, fonksiyona bir argüman olarak geçirildiğini ve `text_to_analyse` değişkeninde saklandığını varsayın.

▼ Çözüm için buraya tıklayın
sentiment_analysis.py

```
import requests  # HTTP isteklerini yönetmek için requests kütüphanesini içe aktar
```

```
def sentiment_analyzer(text_to_analyse):  # Bir dize girişi (text_to_analyse) alan sentiment_analyzer adında bir fonksiyon tanımla
    url = 'https://sn-watson-sentiment-bert.labs.skills.network/v1/watson.runtime.nlp.v1/NlpService/SentimentPredict'  # Duygu analizi hizmetini
    myobj = { "raw_document": { "text": text_to_analyse } }  # Analiz edilecek metin ile bir sözlük oluştur
    header = {"grpc-metadata-mm-model-id": "sentiment_aggregated-bert-workflow_lang_multi_stock"}  # API isteği için gereken başlıkları ayarla
    response = requests.post(url, json = myobj, headers=header)  # Metin ve başlıklarla API'ye POST isteği gönder
    return response.text  # API'den gelen yanıt metnini döndür
```

Bu uygulama artık Python kabuğu kullanılarak çağrılabilir. Uygulamayı test etmek için, mevcut dizin olan `practice_project` içinde Python kabuğunu açmak için python3.11 kullanarak bir Python kabuğu açın. *Mevcut dizinin `practice_project` olduğundan emin olun.*

```
python3.11
```

Python shell'inde `sentiment_analyzer` fonksiyonunu içe aktarın.

▼ İpucu için buraya tıklayın
Sözdizimi:

```
from dosya_adı import fonksiyon_adı
```

▼ Çözüm için buraya tıklayın

```
from sentiment_analysis import sentiment_analyzer
```

Başarılı bir şekilde içe aktardıktan sonra, uygulamanızı "Bu yeni teknolojiyi seviyorum." metniyle test edin.

```
sentiment_analyzer("I love this new technology")
```

Aşağıdaki resimde gösterildiği gibi beklenen sonuç budur. Python kabuğundan çıkmak için `Ctrl+Z` tuşlarına basın veya `exit()` yazın.

Bu, Görev 2'yi tamamlar. Çıktıda bizim için önemli olan bilgilerin yalnızca `label` ve `score` olduğunu unutmayın. Bir sonraki görevde, bu bilgiyi bu çıktılardan çıkaracaksınız.

Görev 3: Uygulamanın çıktısını biçimlendir

Oluşturulan uygulamanın çıktısı bir sözlük biçimindedir, ancak metin olarak biçimlendirilmiştir. Bu çıktıda ilgili bilgilere erişmek için önce bu metni bir sözlüğe dönüştürmemiz gerekiyor. Sözlükler JSON dosyaları için varsayılan biçimlendirme sistemi olduğundan, yerleşik Python kütüphanesi `json`'dan faydalanıyoruz.

Bunun nasıl çalıştığına bakalım.

Öncelikle, bir Python kabuğunda json kütüphanesini içe aktarın.

```
import json
```

Sonraki adımda, “Bu yeni teknolojiyi seviyorum” metni için `sentiment_analyzer` fonksiyonunu, Görev 2’deki gibi çalıştırın ve çıktıyı `response` adlı bir değişkende saklayın.

```
from sentiment_analysis import sentiment_analyzer
response = sentiment_analyzer("I love this new technology")
```

Şimdi `response` değişkenini json.loads fonksiyonuna bir argüman olarak geçirin ve çıktıyı `formatted_response` içinde saklayın. `formatted_response`‘ı yazdırarak formatlamadaki farkı görün.

```
formatted_response = json.loads(response)
print(formatted_response)
```

Yukarıda belirtilen adımların beklenen çıktısı aşağıdaki resimde gösterilmektedir.

Yanıtın her iki tarafında tek tırnakların olmaması, bunun artık bir metin değil, bir sözlük olduğunu gösterir. Bu sözlükten doğru bilgilere erişmek için anahtarları uygun şekilde kullanmamız gerekir. Bu, iç içe geçmiş bir sözlük yapısı olduğu için, yani sözlükler sözlüğü olduğundan, bu yanıtın etiket ve puan çıktısını almak için aşağıdaki ifadelerin kullanılması gerekir.

```
label = formatted_response['documentSentiment']['label']
score = formatted_response['documentSentiment']['score']
```

Kontrol etmek için `label` ve `score` içeriğini doğrulayın.

Şimdi, Görev 3 için, yukarıda bahsedilen tekniği ekleyin ve `sentiment_analysis.py` dosyasında değişiklikler yapın. `sentiment_analyzer` fonksiyonunu çağırdığınızda beklenen çıktı artık 2 anahtara sahip bir sözlük olmalıdır: label ve score, her biri Watson NLP fonksiyonunun yanıtından çıkarılan uygun değeri taşımalıdır. Değişikliklerinizi doğrulamak için değiştirilmiş fonksiyonu bir python kabuğunda test edin.

▼ Çözüm için buraya tıklayın

```
import requests
import json
def sentiment_analyzer(text_to_analyse):
    # Duygu analizi hizmetinin URL'si
    url = 'https://sn-watson-sentiment-bert.labs.skills.network/v1/watson.runtime.nlp.v1/NlpService/SentimentPredict'
    # Beklenen formatta istek yükünü oluşturma
    myobj = { "raw_document": { "text": text_to_analyse } }
    # Duygu analizi hizmeti için model kimliğini belirten özel başlık
    header = {"grpc-metadata-mm-model-id": "sentiment_aggregated-bert-workflow_lang_multi_stock"}
    # Duygu analizi API'sine POST isteği gönderme
    response = requests.post(url, json=myobj, headers=header)
    # API'den gelen JSON yanıtını ayrıştırma
    formatted_response = json.loads(response.text)
    # Yanıttan duygu etiketi ve puanını çıkarma
    label = formatted_response['documentSentiment']['label']
    score = formatted_response['documentSentiment']['score']
    # Duygu analizi sonuçlarını içeren bir sözlük döndürme
    return {'label': label, 'score': score}
```

Tamamlandığında, fonksiyonun beklenen çıktısı aşağıdaki resimde gösterilmektedir.

Python kabuğundan çıkmak için `exit()` yazın veya `Ctrl+Z` tuşuna basın.

Görev 4: Uygulamayı Paketleme

Bu görevde, 2. ve 3. görevlerde oluşturduğunuz son uygulamayı paketlemeniz gerekiyor.

Paketin adını `SentimentAnalysis` olarak koruyalım. Paketleme adımları şunlardır:

1. Çalışma dizininde, paket adıyla aynı isme sahip bir klasör oluşturun.

▼ İpucu için buraya tıklayın
mkdir <package_name>
▼ Çözüm için buraya tıklayın

```
mkdir SentimentAnalysis
```

2. Uygulama kodunu (yani modülü) paket klasörüne taşıyın.

▼ İpucu için buraya tıklayın
`sentiment_analysis.py` dosyasını `SentimentAnalysis` klasörüne taşımak için bir terminal komutu veya Cloud IDE konsolu kullanabilirsiniz.
▼ Çözüm için buraya tıklayın

```
mv ./sentiment_analysis.py ./SentimentAnalysis
```

3. Modülü referans almak için paket klasörünün içinde __init__.py dosyasını oluşturun.

▼ İpucu için buraya tıklayın
Init dosyasında mevcut klasörden modülü/fonksiyonu içe aktarın.
▼ Çözüm için buraya tıklayın
__init__.py dosyasına şu satırı ekleyin:

```
from . import sentiment_analysis
```

Son klasör yapısı aşağıdaki resimde gösterildiği gibi görünmelidir.

`SentimentAnalysis` artık geçerli bir pakettir ve bu projedeki herhangi bir dosyaya içe aktarılabilir.

Bunu test etmek için terminalde bir python shell açın ve paketten `sentiment_analyzer` fonksiyonunu içe aktarmayı deneyin.

▼ İpucu için buraya tıklayın
Bu içe aktarma için sözdizimi

```
from package_name.module_name import function_name
```

▼ Çözüm için buraya tıklayın

```
from SentimentAnalysis.sentiment_analysis import sentiment_analyzer
```

İçe aktarma ifadesinden sonra hata mesajı alınmaması, paketin artık kullanım için hazır olduğu anlamına gelir. Shell'de aşağıdaki ifadeyi çalıştırarak fonksiyonu test edin.

```
sentiment_analyzer("This is fun.")
```

Alınan çıktı aşağıda gösterildiği gibi olacaktır.

Python kabuğundan çıkmak için `exit()` yazın veya `Ctrl+Z` tuşuna basın.

Görev 5: Uygulamanızda Birim Testlerini Çalıştırın

Artık işlevsel bir uygulamaya sahip olduğumuza göre, çıktılarının geçerliliğini kontrol etmek için bazı test durumlarında birim testlerini çalıştırmamız gerekmektedir.

Birim testlerini çalıştırmak için, paketten gerekli uygulama fonksiyonunu çağıran ve bilinen bir metin ile çıktı çiftini test eden yeni bir dosya oluşturmamız gerekiyor.

Bunun için aşağıdaki adımları tamamlayın.

1. `practice_project` klasöründe `test_sentiment_analysis.py` adında yeni bir dosya oluşturun.
2. Bu dosyada, `SentimentAnalysis` paketinden `sentiment_analyzer` fonksiyonunu içe aktarın. Ayrıca `unittest` kütüphanesini de içe aktarın.

▼ Çözüm için buraya tıklayın

```
from SentimentAnalysis.sentiment_analysis import sentiment_analyzer
import unittest
```

3. Birim test sınıfını oluşturun. Buna TestSentimentAnalyzer diyelim. Birim testlerini çalıştıracak fonksiyon olarak `test_sentiment_analyzer` tanımlayın.

▼ Çözüm için buraya tıklayın

```
class TestSentimentAnalyzer(unittest.TestCase):
    def test_sentiment_analyzer(self):
```

4. Belirtilen fonksiyonda 3 birim testi tanımlayın ve aşağıdaki ifade - etiket çiftinin geçerliliğini kontrol edin.
   "Python ile çalışmayı seviyorum": "SENT_POSITIVE"
   "Python ile çalışmayı sevmiyorum": "SENT_NEGATIVE"
   "Python hakkında tarafsızım": "SENT_NEUTRAL"

▼ İpucu için buraya tıklayın
`label` çıktısını beklenen etiket ile karşılaştırmak için `assertEqual` fonksiyonunu kullanın.
▼ Çözüm için buraya tıklayın

```
class TestSentimentAnalyzer(unittest.TestCase):
    def test_sentiment_analyzer(self):
        # Olumlu duygu için test durumu
        result_1 = sentiment_analyzer('I love working with Python')
        self.assertEqual(result_1['label'], 'SENT_POSITIVE')
        # Olumsuz duygu için test durumu
        result_2 = sentiment_analyzer('I hate working with Python')
```

```
        self.assertEqual(result_2['label'], 'SENT_NEGATIVE')
        # Tarafsız duygu için test durumu
        result_3 = sentiment_analyzer('I am neutral on Python')
        self.assertEqual(result_3['label'], 'SENT_NEUTRAL')
```

5. Birim testlerini çağırın.

▼ Çözüm için buraya tıklayın
Dosyanın sonuna aşağıdaki satırı ekleyin.

```
unittest.main()
```

Artık dosya hazır olduğuna göre, birim testlerini gerçekleştirmek için dosyayı çalıştırın. Başarılı bir şekilde çalıştırıldığında, bu dosyanın çıktısı aşağıdaki görüntüde gösterildiği gibi olmalıdır.

Görev 6: Flask Kullanarak Web Uygulaması Olarak Yayınla

Uygulama hazır olduğuna göre, artık web arayüzü üzerinden kullanım için onu dağıtma zamanı. Dağıtım sürecini kolaylaştırmak için bu görev için kullanılacak 3 dosya sağlanmıştır.

- Dizin yapısını doğrulayın:

```
practice_project/
├── SentimentAnalysis/
│   ├── __init__.py
│   ├── sentiment_analysis.py
├── templates/
│   ├── index.html
├── static/
│   ├── mywebscript.js
├── server.py
```

- `templates` klasöründeki bu `index.html` dosyası, bu laboratuvar için tasarlanmış web arayüzünün kodunu içermektedir. Bu dosya tamamlanmış bir şekilde size sağlanmakta ve olduğu gibi kullanılmalıdır. Bu dosyada herhangi bir değişiklik yapmanız gerekmemektedir.

Arayüz aşağıdaki resimde gösterilmiştir.

- HTML arayüzündeki `Duygu Analizini Çalıştır` butonuna tıklandığında, `static` klasöründeki bu mywebscript.js JavaScript dosyasını çağırır. Bu dosya, kullanıcı tarafından sağlanan metni girdi olarak alarak bir GET isteği gerçekleştirir. Bu metin, `textToAnalyze` adında bir değişkende saklanır ve ardından sunucu dosyasına iletilir. Bu dosya da tamamlanmış bir şekilde size sağlanmakta ve olduğu gibi kullanılmalıdır. Bu dosyada herhangi bir değişiklik yapmanız gerekmemektedir.

- `practice_project` klasöründeki `server.py` dosyasını açın. Bu görev, bu dosyanın tamamlanması etrafında döner. Bu dosyayı aşağıdaki 5 adımı tamamlayarak tamamlayabilirsiniz.

a. *İlgili kütüphaneleri ve fonksiyonları içe aktarın*

Bu dosyada, HTML dosyasını dağıtmak için `render_template` fonksiyonu ile birlikte Flask kütüphanesine ve web sayfasından GET isteğini başlatmak için `request` fonksiyonuna ihtiyacınız olacak.

Ayrıca `SentimentAnalysis` paketinden `sentiment_analyzer` fonksiyonunu da içe aktarmanız gerekecek.

Belirtilen fonksiyonları içe aktaran ilgili kod satırlarını `server.py` dosyasına ekleyin.

▼ Çözüm için buraya tıklayın

```
from flask import Flask, render_template, request
from SentimentAnalysis.sentiment_analysis import sentiment_analyzer
```

b. *Flask uygulamasını `Sentiment Analyzer` adıyla başlatın*

Bu kursun 2. Modülünde edindiğiniz bilgileri kullanarak, uygulamayı başlatan ve adını `Sentiment Analyzer` olarak belirten ifadeyi server.py dosyasına ekleyin.

▼ Çözüm için buraya tıklayın

```
app = Flask("Sentiment Analyzer")
```

c. *`sent_analyzer` fonksiyonunu tanımlayın*

Bu fonksiyonun amacı iki yönlüdür. İlk olarak, fonksiyon HTML arayüzüne bir GET isteği göndererek girdi metnini almalıdır. GET isteğinin `mywebscript.js` dosyasında tanımlanan `textToAnalyze` değişkenine atıfta bulunması gerektiğini unutmayın. Gelen metni `text_to_analyze` adında bir değişkende saklayın. İkinci olarak, `text_to_analyze` argümanı ile `sentiment_analyzer` uygulamanızı çağırın.

Ayrıca, fonksiyonun dönen çıktısını resmi bir metin formatında düzenleyin. Örneğin:
`Verilen metin POSİTİF olarak 0.99765 puanıyla tanımlanmıştır.`

▼ İpucu için buraya tıklayın

1. GET isteğini başlatmak için request.args.get kullanın.
2. "SENT_CLASS" olarak alınan etiket (burada sınıf POSİTİF, NEGATİF veya NÖTR olabilir), sınıf adını bireysel olarak erişmek için '_' ile bölünmelidir.

▼ Çözüm için buraya tıklayın
Fonksiyon aşağıdaki gibi olmalıdır.

```
@app.route("/sentimentAnalyzer")
def sent_analyzer():
    # İstek argümanlarından analiz edilecek metni alın
    text_to_analyze = request.args.get('textToAnalyze')
    # Metni sentiment_analyzer fonksiyonuna iletin ve yanıtı saklayın
    response = sentiment_analyzer(text_to_analyze)
    # Yanıttan etiket ve puanı çıkarın
    label = response['label']
    score = response['score']
    # Duygu etiketi ve puanıyla birlikte biçimlendirilmiş bir dize döndürün
    return "Verilen metin {} olarak tanımlanmıştır ve puanı {}'dir.".format(label.split('_')[1], score)
```

Not: Fonksiyon, `mywebscript.js` dosyasında referans verilen Flask dekoratörü `@app.route("/sentimentAnalyzer")` kullanmaktadır.

d. *HTML şablonunu `render_index_page` ile render edin*

Bu fonksiyon, `index.html` HTML şablonunda render_template fonksiyonunu çalıştırmalıdır.

▼ Çözüm için buraya tıklayın

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

e. *Uygulamayı `localhost:5000` üzerinde çalıştırın*

Son olarak, dosya çalıştırıldığında, uygulamayı `0.0.0.0` (veya localhost) üzerinde 5000 port numarasıyla çalıştırın.

▼ Çözüm için buraya tıklayın

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Uygulamayı dağıtmak için, terminalden *server.py* dosyasını çalıştırın.

```
python3.11 server.py
```

Çıktı şu şekilde görünecektir.

Uygulama artık localhost:5000 üzerinde çalışıyor. Uygulamaya erişmek için Skills Network Toolbox sekmesine gidin ve Uygulamayı Başlat'a tıklayın. Uygulama portunu 5000 olarak girin ve Uygulamanız'a tıklayın.

Uygulama arayüzü açılacaktır. Uygulamanızı test etmek için arayüzü kullanın.

Uygulamayı durdurmak için `Ctrl+C` tuşlarına basın.

Görev 7: Hata Yönetimini Dahil Etme

Hata yönetimini dahil etmek için, `server.py` içindeki `sent_analyzer` fonksiyonu tarafından başlatılan GET sorgusuna yanıt olarak alınabilecek farklı hata kodlarını tanımlamamız gerekiyor.

Bu, Watson NLP Kütüphanesi fonksiyonlarının bir parçasıdır ve kodun çalıştığı terminal konsolunda gözlemlenebilir.

Aşağıda gösterilen görüntüyü dikkate alın.

Kodlar, ilk GET isteğinin başarılı olduğunu (200), isteğin Watson Kütüphanesine başarıyla aktarıldığını (304) ve ardından yanıtı oluşturmak için GET isteğinin de başarıyla gerçekleştirildiğini göstermektedir.

Geçersiz girişler durumunda, sistem 500 hata kodu ile yanıt verir ve bunun sunucu tarafında bir sorun olduğunu belirtir.
Geçersiz giriş, modelin yorumlayamadığı herhangi bir şey olabilir. Ancak, bu hata durumunda, bu uygulamanın çıktısı güncellenmez.

Arayüzdeki çıktının öncekiyle aynı olduğunu, analiz edilen metnin rastgele bir metin olduğunu ve Watson AI kütüphanelerinin, modelin isteği işleyemediğini doğrulayan 500 hata kodu verdiğini unutmayın.

Uygulamamızdaki bu hatayı düzeltmek için, sunucu 500 hata kodu ürettiğinde Watson AI kütüphane fonksiyonundan alınan yanıtı incelememiz gerekiyor. Bunu test etmek için, Görev 2'de atılan adımları geri takip etmemiz ve Watson AI kütüphanesini geçersiz bir dize girişi ile test etmemiz gerekecek.

Terminalde bir Python kabuğu açın ve aşağıdaki komutları çalıştırarak sentiment_analysis.py dosyasını güncelledikten sonra gereken çıktıyı kontrol edin.

```
import requests
# Define the URL for the sentiment analysis API
url = "https://sn-watson-sentiment-bert.labs.skills.network/v1/watson.runtime.nlp.v1/NlpService/SentimentPredict"
# Set the headers with the required model ID for the API
headers = {"grpc-metadata-mm-model-id": "sentiment_aggregated-bert-workflow_lang_multi_stock"}
# Define the first payload with nonsensical text to test the API
myobj = { "raw_document": { "text": "as987da-6s2d aweadsa" } }
# Make a POST request to the API with the first payload and headers
response = requests.post(url, json=myobj, headers=headers)
# Print the status code of the first response
print(response.status_code)
# Define the second payload with a meaningful text to test the API
myobj = { "raw_document": { "text": "Testing this application for error handling" } }
# Make a POST request to the API with the second payload and headers
response = requests.post(url, json=myobj, headers=headers)
# Print the status code of the second response
print(response.status_code)
```

Konsol yanıtı aşağıdaki resimde gösterildiği gibidir. Kırmızı kutular geçersiz metni ve alınan durum kodunu, sarı kutular ise geçerli metni ve alınan durum kodunu göstermektedir.

Bu, uygulamayı öyle bir şekilde değiştirmenizi sağlar ki, farklı durum kodları için farklı çıktılar gönderebiliriz.

Bu görevin ilk kısmında, geçersiz metin girişi durumunda `label` ve `score` değerlerini `None` döndürecek şekilde `sentiment_analyzer()` fonksiyonunu değiştirmeniz gerekiyor.

▼ İpucu için buraya tıklayın
Gerekli işlevselliği eklemek için sentiment_analyzer fonksiyonunda bir if-else koşul ifadesi oluşturun.
▼ Çözüm için buraya tıklayın
sentiment_analysis.py

```
import requests
import json
def sentiment_analyzer(text_to_analyse):
    # Duygu analizi API'si için URL'yi tanımlayın
    url = 'https://sn-watson-sentiment-bert.labs.skills.network/v1/watson.runtime.nlp.v1/NlpService/SentimentPredict'
    # Analiz edilecek metni içeren yükü oluşturun
    myobj = { "raw_document": { "text": text_to_analyse } }
    # API için gerekli model ID'si ile başlıkları ayarlayın
    header = {"grpc-metadata-mm-model-id": "sentiment_aggregated-bert-workflow_lang_multi_stock"}
    # Yük ve başlıklarla API'ye POST isteği gönderin
    response = requests.post(url, json=myobj, headers=header)
    # API'den gelen yanıtı ayrıştırın
    formatted_response = json.loads(response.text)
    # Yanıt durum kodu 200 ise, yanıtın etiketini ve puanını çıkarın
    if response.status_code == 200:
        label = formatted_response['documentSentiment']['label']
        score = formatted_response['documentSentiment']['score']
    # Yanıt durum kodu 500 ise, label ve score'u None olarak ayarlayın
    elif response.status_code == 500:
        label = None
        score = None
    # Etiket ve puanı bir sözlükte döndürün
    return {'label': label, 'score': score}
```

Şimdi, `server.py` dosyasında konsola gönderilecek yanıt da geçerli ve geçersiz giriş türleri için farklı olmalıdır. Geçersiz giriş için, konsolun `Invalid input ! Try again.` yazdırmasını sağlayın.

▼ İpucu için buraya tıklayın
server.py içindeki sent_analyzer fonksiyonunda "label" değerinin None olup olmadığını kontrol etmek için bir if-else koşul ifadesi oluşturun.
▼ Çözüm için buraya tıklayın

```
def sent_analyzer():
    # İstek argümanlarından analiz edilecek metni alın
    text_to_analyze = request.args.get('textToAnalyze')
    # Metni sentiment_analyzer fonksiyonuna iletin ve yanıtı saklayın
    response = sentiment_analyzer(text_to_analyze)
    # Yanıttan etiket ve puanı çıkarın
    label = response['label']
    score = response['score']
    # Etiketin None olup olmadığını kontrol edin, bu bir hata veya geçersiz girişi gösterir
    if label is None:
        return "Geçersiz giriş! Tekrar deneyin."
    else:
        # Duygu etiketini ve puanını içeren biçimlendirilmiş bir dize döndürün
        return "Verilen metin {} olarak tanımlandı ve puanı {}.".format(label.split('_')[1], score)
```

Artık uygulamanız, her türlü girişe uygun şekilde yanıt verebilmektedir.

Görev 8: Statik kod analizi yapın

Son olarak, Görev 8'de, PEP8 yönergelerine göre kodlama becerilerinizin kalitesini statik kod analizi yaparak kontrol ediyoruz.

Normalde, bu işlem uygulamanın paketlenmesi ve birim testleri sırasında yapılır. Ancak, bu projede bu adımı en sona koyduk çünkü kodlar, bu aşamadan önceki tüm görevlerde güncellendi. Proje dosyalarınız şimdi hazır olduğuna göre, PEP8 yönergelerine uyum açısından test edelim.

Bu süreçteki ilk adım, terminal kullanarak `PyLint` kütüphanesini yüklemektir.

▼ Çözüm için buraya tıklayın

```
python3.11 -m pip install pylint
```

Sonrasında, `server.py` üzerinde statik kod analizi yapmak için pylint'i kullanın.

▼ Çözüm için buraya tıklayın
Terminal bash'te aşağıdaki komutu çalıştırın.

```
pylint server.py
```

Eğer kodunuzda PEP8 kılavuzunun tüm yönleri yer alıyorsa, üretilen puan 10/10 olmalıdır. Eğer bu böyle değilse, kodu uygun şekilde düzeltmek için pylint kütüphanesinin verdiği talimatları izleyin.

Bu, uygulama projesinin sonunu oluşturur.

(İsteğe Bağlı) Ek Alıştırmalar

İlgilenen öğreniciler, bu projede öğrenilen kavramların daha iyi anlaşılması için aşağıdaki alıştırmaları kendi başlarına deneyebilirler. Bu alıştırmalar için çözüm sağlanmamaktadır. Ancak, çözümlerinizi kurs tartışma forumlarında arkadaşlarınızla tartışmaktan ve paylaşmaktan çekinmeyin.

1. `sentiment_analysis.py` üzerinde statik kod analizi yapın. 10/10 puan almaya çalışın. İpucu: *Docstring'ler*
2. Uygulamanızın İngilizce dışındaki dillerdeki cümleleri işleme kapasitesini test edin; örneğin, Fransızca, Almanca vb. Uygulamanın geçersiz metin hatası verip vermediğine bakın.
3. Şu anda, uygulama bir giriş sağlamadan çalıştırıldığında, yani metni boş bıraktığınızda, model yine geçersiz metin hatası veriyor. Boş bir girişin farklı bir hata mesajı alacağı özel bir durumu dahil etmeyi deneyin.

Sonuç

Bu projeyi tamamladığınız için tebrikler.

Bu projenin tamamlanmasıyla birlikte:

1. Watson NLP gömülü kütüphanelerini kullanarak bir AI tabanlı duygu analizi uygulaması oluşturmuş oldunuz.
2. Watson NLP kütüphanesi fonksiyonundan alınan çıktıyı, içinden ilgili bilgileri çıkartacak şekilde biçimlendirdiniz.
3. Uygulamayı paketleyip, kullanım için herhangi bir Python koduna dahil edilebilir hale getirdiniz.
4. Uygulama üzerinde birim testleri çalıştırarak, farklı girdiler için çıktılarının geçerliliğini kontrol ettiniz.
5. Uygulamayı Flask framework'ü kullanarak dağıttınız.
6. Uygulamada hata yönetimi yeteneği ekleyerek, 500 yanıt kodunun uygulamadan uygun bir yanıt almasını sağladınız.
7. Kod dosyalarında statik kod analizi yaparak, PEP8 yönergelerine uyumlarını onayladınız.