

Abschlussprojekt: Entwicklung und Bereitstellung einer KI-basierten Webanwendung



Geschätzte Zeit: 1 Stunde 45 Minuten

Szenario

Sie wurden von einem E-Commerce-Unternehmen als Softwareentwickler eingestellt, um eine KI-basierte Webanwendung zu erstellen, die Analysen zu Kundenfeedback für deren Hauptprodukte durchführt. Um dieses Ziel zu erreichen, werden Sie ein Emotionserkennungssystem erstellen, das das Feedback, das der Kunde in Textform bereitstellt, verarbeitet und die damit verbundene Emotion entschlüsselt.

Einführung

In diesem Abschlussprojekt werden Sie auf das Wissen bewertet, das Sie in allen Aspekten der App-Erstellung und deren Web-Bereitstellung während dieses Kurses erworben haben. Sie müssen von Zeit zu Zeit Screenshots Ihrer Ergebnisse mit spezifischer Nomenklatur speichern. Diese Screenshots müssen in der anschließenden peer-bewerteten Aufgabe hochgeladen werden.

In diesem Projekt verwenden wir die einbettbaren Watson AI-Bibliotheken, um eine Emotionserkennungsanwendung zu erstellen.

Die Emotionserkennung erweitert das Konzept der Sentimentanalyse, indem sie die feineren Emotionen wie Freude, Traurigkeit, Wut usw. aus Aussagen extrahiert, anstatt nur die einfache Polarität, die die Sentimentanalyse bietet. Dies macht die Emotionserkennung zu einem sehr wichtigen Studienbereich, und Unternehmen nutzen solche Systeme weitreichend für ihre KI-basierten Empfehlungssysteme, automatisierte Chatbots usw.

Projektleitlinien

Für den Abschluss dieses Projekts müssen Sie die folgenden 8 Aufgaben basierend auf dem Wissen, das Sie im Laufe des Kurses erworben haben, abschließen.

Hinweis: Diese Plattform ist nicht persistent. Es wird empfohlen, eine Kopie Ihres Codes auf Ihren lokalen Maschinen zu behalten und von Zeit zu Zeit Änderungen zu speichern. Falls Sie das Labor erneut besuchen, müssen Sie die Dateien in dieser Laborumgebung erneut erstellen, indem Sie die gespeicherten Kopien von Ihren Maschinen verwenden.

Aufgaben und Ziele:

- Aufgabe 1: Projekt-Repository forken und klonen
- Aufgabe 2: Eine Emotionserkennungsanwendung mit der Watson NLP-Bibliothek erstellen
- Aufgabe 3: Die Ausgabe der Anwendung formatieren
- Aufgabe 4: Die Anwendung paketieren
- Aufgabe 5: Unit-Tests Ihrer Anwendung ausführen
- Aufgabe 6: Als Webanwendung mit Flask bereitstellen
- Aufgabe 7: Fehlerbehandlung einfügen
- Aufgabe 8: Statische Codeanalyse durchführen

Lassen Sie uns anfangen!

Aufgabe 1: Forken und Klonen des Projekt-Repositorys

Hinweis: Bitte führen Sie dieses Labor in der Skills Network Theia Lab-Umgebung selbst aus, nicht in Ihrer lokalen IDE (wie VS Code). Die hier verwendete API wird auf der Skills Network-Plattform gehostet und ist nur innerhalb des Theia Labs zugänglich.

Das GitHub-Repository des Projekts ist unter der unten angegebenen URL verfügbar.

Als ersten Schritt müssen Sie dieses Repository forken, klicken Sie auf die Schaltfläche “Fork” in der oberen rechten Ecke der Repository-Seite.

<https://github.com/ibm-developer-skills-network/oajjp-final-project-emb-ai.git>

Sie finden Anweisungen zum Forken des Repositories, indem Sie Übung 2 unter folgendem [Link](#) besuchen.

Hinweis: Stellen Sie sicher, dass Ihr geforktes Repository öffentlich ist. Diese Aktion erstellt eine Kopie des geforkten Repositories in Ihrem GitHub-Konto.

1. Öffnen Sie ein neues Terminal und erstellen Sie das Verzeichnis `final_project` mit dem Befehl `mkdir`.
2. Klonen Sie dieses geforkte GitHub-Repository über das Cloud IDE-Terminal in einen Ordner namens `final_project`.
Sie finden Anweisungen, wie Sie die Klon-URL des Repositories erhalten, indem Sie Übung 3 unter folgendem [Link](#) besuchen.
3. Nachdem das Klonen abgeschlossen ist, verwenden Sie das Terminal, um das aktuelle Verzeichnis auf `final_project` zu ändern.
4. Machen Sie einen **Screenshot** Ihrer Ordnerstruktur. Speichern Sie dieses Bild als `1_folder_structure.png`.

Aufgabe 2: Erstellen Sie eine Anwendung zur Emotionserkennung mit der Watson NLP-Bibliothek

Die Watson NLP-Bibliotheken sind eingebettet. Daher ist es nicht erforderlich, sie in Ihren Code zu importieren. Sie müssen lediglich eine POST-Anfrage an die richtige Funktion in der Bibliothek senden und die Ausgabe erhalten.

1. Für dieses Projekt verwenden Sie die **Emotion Predict**-Funktion der Watson NLP-Bibliothek. Um auf diese Funktion zuzugreifen, sind die URL, die Header und das Eingabe-JSON-Format wie folgt.

```
URL: 'https://sn-watson-emotion.labs.skills.network/v1/watson.runtime.nlp.v1/NlpService/EmotionPredict'  
Headers: {"grpc-metadata-mm-model-id": "emotion_aggregated-workflow_lang_en_stock"}  
Input json: { "raw_document": { "text": text_to_analyse } }
```

Beachten Sie, dass `text_to_analyse` als Variable verwendet wird, die den tatsächlich zu analysierenden Text enthält.

2. Erstellen Sie eine Datei mit dem Namen `emotion_detection.py` im Ordner `final_project`.
3. Schreiben Sie in die Datei `emotion_detection.py` die Funktion zur Durchführung der Emotionserkennung mit der entsprechenden Funktion zur Emotionserkennung. Nennen Sie diese Funktion `emotion_detector`.

Hinweis: Gehen Sie davon aus, dass der zu analysierende Text als Argument an die Funktion übergeben wird und in der Variable `text_to_analyse` gespeichert ist. Der zurückgegebene Wert muss das Attribut `text` des Antwortobjekts sein, das von der Funktion zur Emotionserkennung empfangen wurde.

4. Machen Sie einen **Screenshot** des Codes, den Sie geschrieben haben, und speichern Sie ihn als `2a_emotion_detection.png`.
5. Die Anwendung sollte jetzt mit der Python-Shell importierbar sein. Öffnen Sie eine `python3`-Shell.
6. Importieren Sie die Anwendung.
7. Nachdem Sie die Anwendung erfolgreich importiert haben, testen Sie Ihre Anwendung mit dem Text: "**Ich liebe diese neue Technologie.**"
8. Machen Sie einen **Screenshot** des Terminals, der alle drei Schritte zusammen mit der endgültigen Ausgabe enthält. Nennen Sie diese Datei `2b_application_creation.png`.

Hinweis: Falls die Python-Shell einen Fehler `ModuleNotFoundError: No module named 'requests'` anzeigt, können Sie die `requests`-Bibliothek mit dem folgenden Befehl im Terminal in Ihre IDE installieren.

```
python3 -m pip install requests
```

Optional:

Zu jedem Zeitpunkt, wenn Sie Ihren Code in Ihr geforktes GitHub-Repository pushen möchten, können Sie dies tun, indem Sie die Anweisungen in diesem [link](#) befolgen.

Aufgabe 3: Formatieren Sie die Ausgabe der Anwendung

1. Konvertieren Sie den Antworttext in ein Dictionary mithilfe der Funktionen der `json`-Bibliothek.
Beachten Sie die Inhaltsformatierung in diesem Dictionary.
2. Extrahieren Sie die erforderliche Menge an Emotionen, einschließlich Wut, Ekel, Angst, Freude und Traurigkeit, zusammen mit ihren Punktzahlen.
3. Schreiben Sie die Logik, um die dominante Emotion zu finden, die die Emotion mit der höchsten Punktzahl ist.
4. Modifizieren Sie dann die Funktion `emotion_detector`, um das folgende Ausgabeformat zurückzugeben.

```
{
  'anger': anger_score,
  'disgust': disgust_score,
  'fear': fear_score,
  'joy': joy_score,
  'sadness': sadness_score,
  'dominant_emotion': '<name of the dominant emotion>'
}
```

5. Machen Sie einen **Screenshot** des modifizierten Funktionscodes und speichern Sie ihn als `3a_output_formatting.png`.
6. Testen Sie die Anwendung erneut im `python3`-Shell mit der Aussage **Ich bin so glücklich, dass ich das mache.**
7. Überprüfen Sie, ob die empfangene Antwort die dominante Emotion **Freude** hat.

8. Machen Sie einen **Screenshot** der Ausgabe aus dem Terminal-Shell und speichern Sie ihn als `3b_formatted_output_test.png`.

Optional

Wenn Sie zu irgendeinem Zeitpunkt Ihren Code in Ihr geforktes GitHub-Repository pushen möchten, können Sie dies tun, indem Sie die Anweisungen in diesem [Link](#) befolgen.

Aufgabe 4: Anwendung paketieren

In dieser Aufgabe werden Sie die in den vorherigen Schritten erstellte Anwendung paketieren.

1. Führen Sie die relevanten Schritte zur Erstellung des Pakets aus. Setzen Sie den Namen des Pakets auf `EmotionDetection`.
2. Sobald dies abgeschlossen ist, machen Sie einen **Screenshot** des Inhalts der `init`-Datei sowie der endgültigen Ordnerstruktur des `final_project`-Verzeichnisses.

Stellen Sie sicher, dass beide Bilder in einen einzigen Screenshot passen und benennen Sie ihn `4a_packaging.png`.

3. Um sicherzustellen, dass `EmotionDetection` ein gültiges Paket ist, müssen Sie das Paket testen. Gehen Sie dazu wie folgt vor:
 1. Führen Sie eine Python-Shell im Terminal aus.
 2. Importieren Sie die `emotion_detector`-Funktion aus dem Paket.
 3. Wenn das Paket korrekt erstellt wurde, erhalten Sie keine Fehlermeldungen.
 4. Führen Sie die Funktion erneut mit der Aussage **Ich hasse es, lange Stunden zu arbeiten** aus.
 5. Überprüfen Sie, ob die Ausgabe die dominante Emotion als **Wut** anzeigt.
4. Machen Sie einen **Screenshot** dieser Terminalausgabe und speichern Sie ihn als `4b_packaging_test.png`.

Optional:

Wenn Sie zu irgendeinem Zeitpunkt Ihren Code in Ihr geforktes GitHub-Repository pushen möchten, können Sie dies tun, indem Sie die Anweisungen in diesem [Link](#) befolgen.

Aufgabe 5: Führen Sie Unit-Tests für Ihre Anwendung aus

Mit einer funktionalen Anwendung, die verfügbar ist, müssen Sie nun Unit-Tests durchführen, um die Ausgabe der Anwendung zu validieren.

1. Um Unit-Tests auszuführen, erstellen Sie eine neue Datei, `test_emotion_detection.py`, die die erforderliche Anwendungsfunktion aus dem Paket aufruft und sie für die folgenden Aussagen und dominanten Emotionen testet.

Aussage	Dominante Emotion
Ich bin froh, dass das passiert ist	Freude
Ich bin wirklich wütend darüber	Wut
Ich fühle mich schon beim Hören davon angewidert	Ekel
Ich bin so traurig darüber	Traurigkeit
Ich habe wirklich Angst, dass das passieren wird	Angst

2. Sobald Sie fertig sind, machen Sie einen **Screenshot** des Codes und speichern Sie ihn als `5a_unit_testing.png`.
3. Führen Sie die Datei `test_emotion_detection.py` im Terminal aus.

4. Überprüfen Sie die Ausgabe der Unit-Tests, um zu verifizieren, dass die Unit-Tests bestanden wurden.

5. Machen Sie einen **Screenshot** der Terminalausgabe dieser Ausführung und speichern Sie ihn als `5b_unit_testing_result.png`.

Optional:

Wenn Sie zu irgendeinem Zeitpunkt Ihren Code in Ihr geforktes GitHub-Repository pushen möchten, können Sie dies tun, indem Sie die Anweisungen in diesem [Link](#) befolgen.

Aufgabe 6: Webbereitstellung der Anwendung mit Flask

Nachdem alle Tests erfolgreich abgeschlossen sind, müssen Sie die Anwendung im Web bereitstellen. Dies ermöglicht es dem Kunden, auf die Anwendung zuzugreifen und sie zu nutzen.

Hinweis: Die `index.html`-Datei im `templates`-Ordner und die `mywebscript.js`-Datei im `static`-Ordner wurden als Teil des Repos bereitgestellt. Diese Dateien müssen in diesem Projekt nicht aktualisiert werden.

1. Sie müssen die Datei `server.py` von Grund auf neu erstellen. Beachten Sie die folgenden Anforderungen dieser Aufgabe.

Hinweis: `server.py` ist Teil des `final_project`-Ordners.

2. Stellen Sie sicher, dass der Flask-Dekorator für die Anwendungaufruffunktion `\emotionDetector` ist.

3. Der Kunde hat darum gebeten, dass die Ausgabe im Format angezeigt wird, wie im folgenden Beispiel gezeigt.

Beispielausgabe

Angenommen, Sie möchten die Aussage `Ich liebe mein Leben` bewerten. Die Aussage wird wie folgt verarbeitet:

```
{  
    "anger": 0.006274985,  
    "disgust": 0.0025598293,  
    "fear": 0.009251528,  
    "joy": 0.9680386,  
    "sadness": 0.049744144,  
    "dominant_emotion": "joy"  
}
```

Die Antwort wird angezeigt als:

Für die gegebene Aussage lautet die Systemantwort 'anger': 0.006274985, 'disgust': 0.0025598293, 'fear': 0.009251528, 'joy': 0.9680386 und 'sadness': 0.049744144. Die dominante Emotion ist **joy**.

4. Die Anwendung muss auf `localhost:5000` bereitgestellt werden.

Um diese Aufgabe des Projekts abzuschließen, müssen Sie zunächst einen Screenshot des endgültigen Inhalts von `server.py` machen und ihn `6a_server.png` nennen. Zweitens müssen Sie einen Screenshot der endgültig bereitgestellten Anwendung machen, indem Sie sie mit der Aussage "Ich glaube, ich habe Spaß" testen. Nennen Sie dieses Bild `6b_deployment_test.png`.

Optional:

Wenn Sie zu irgendeinem Zeitpunkt Ihren Code in Ihr geforktes GitHub-Repository pushen möchten, können Sie dies tun, indem Sie die Anweisungen in diesem [Link](#) befolgen.

Aufgabe 7: Fehlerbehandlung einfügen

Fügen Sie die Fehlerbehandlungsfunktionalität in Ihre Funktion `emotion_detector` ein, um leere Eingaben von Benutzern zu verwalten, d.h. die Anwendung ohne Eingabe auszuführen.

1. Greifen Sie auf das Attribut `status_code` der Serverantwort zu, um die Systemantwort für leere Eingaben korrekt anzuzeigen.

Für `status_code = 400` sollte die Funktion dasselbe Dictionary zurückgeben, jedoch mit Werten für alle Schlüssel, die `None` sind.

2. Machen Sie einen **Screenshot** der modifizierten Funktion und benennen Sie ihn `7a_error_handling_function.png`.
3. Ändern Sie `server.py`, um die Fehlerbehandlung zu integrieren, wenn `dominant_emotion None` ist. In diesem Szenario sollte die Antwort eine Nachricht anzeigen: **Ungültiger Text! Bitte versuchen Sie es erneut!**.
4. Machen Sie einen **Screenshot** dieser bearbeiteten Datei und speichern Sie ihn als `7b_error_handling_server.png`.
5. Stellen Sie die Anwendung bereit und testen Sie sie auf leere Eingaben.
6. Die Ausgabe sollte die Fehlermeldung anzeigen: **Ungültiger Text! Bitte versuchen Sie es erneut!**.
7. Machen Sie einen **Screenshot** der Ausgabe, die die Fehlermeldung anzeigt, und benennen Sie ihn `7c_error_handling_interface.png`.

Optional:

Wenn Sie zu irgendeinem Zeitpunkt Ihren Code in Ihr geforktes GitHub-Repository pushen möchten, können Sie dies tun, indem Sie die Anweisungen in diesem [Link](#) folgen.

Aufgabe 8: Führen Sie eine statische Codeanalyse durch

Schließlich ist es Zeit, die Einhaltung des Codes zu überprüfen.

1. Führen Sie eine statische Codeanalyse des Codes durch, den Sie erstellt haben. Verwenden Sie die PyLint-Bibliothek für die Datei `server.py` und versuchen Sie, eine Punktzahl von 10/10 zu erreichen. Sie müssen die Datei `server.py` ändern, um diese Punktzahl zu erhalten. Machen Sie einen Screenshot der modifizierten Datei und benennen Sie ihn `8a_server_modified.png`.

Hinweis: Der Unterschied zwischen einer Punktzahl von 9/10 und 10/10 kann die Verwendung von Docstrings sein. Fügen Sie Docstrings in allen Funktionen Ihres Codes hinzu.

2. Machen Sie einen **Screenshot** der 10/10-Ausgabe im Terminal nach der Durchführung der statischen Codeanalyse. Benennen Sie den Screenshot `8b_static_code_analysis.png`.

Checkliste für die Bilder

Bevor Sie das Projekt zur Bewertung einreichen, stellen Sie sicher, dass Sie alle Bilder wie während des Projekts angewiesen erfasst haben. Hier ist eine schnelle Zusammenfassung der Anforderungen.

Aufgabe 1: Klonen Sie das Projekt-Repository

`1_folder_structure.png`

Aufgabe 2: Erstellen Sie eine Anwendung zur Emotionserkennung mit der Watson NLP-Bibliothek

`2a_emotion_detection.png`
`2b_application_creation.png`

Aufgabe 3: Formatieren Sie die Ausgabe der Anwendung

`3a_output_formatting.png`
`3b_formatted_output_test.png`

Aufgabe 4: Verpacken Sie die Anwendung

`4a_packaging.png`
`4b_packaging_test.png`

Aufgabe 5: Führen Sie Unit-Tests für Ihre Anwendung durch

5a_unit_testing.png
5b_unit_testing_result.png

Aufgabe 6: Bereitstellung als Webanwendung mit Flask

6a_server.png
6b_deployment_test.png

Aufgabe 7: Fehlerbehandlung integrieren

7a_error_handling_function.png
7b_error_handling_server.png
7c_error_handling_interface.png

Aufgabe 8: Führen Sie eine statische Codeanalyse durch

8a_server_modified.png
8b_static_code_analysis.png

Optional:

Wenn Sie Ihren Code in Ihr geforktes GitHub-Repository pushen möchten, können Sie dies tun, indem Sie die Anweisungen in diesem [Link](#) befolgen. Dies stellt sicher, dass Sie Ihre Arbeit jederzeit leicht überprüfen können.

Fazit

Herzlichen Glückwunsch zum Abschluss des Projekts.

Durch den Abschluss dieses Projekts haben Sie:

1. Eine Anwendung zur Emotionserkennung unter Verwendung der Funktionen aus einbettbaren KI-Bibliotheken erstellt
2. Relevante Informationen aus den Ausgaben der Funktion extrahiert
3. Die Anwendung, die mit der Funktion zur Emotionserkennung erstellt wurde, getestet und verpackt
4. Die Webbereitstellung Ihrer Anwendung mit Flask abgeschlossen
5. Fehlerbehandlung in Ihre Anwendung integriert, um ungültige Eingaben zu berücksichtigen
6. Codes geschrieben, die vollständig den PEP8-Richtlinien entsprechen und in der statischen Codeanalyse eine Bewertung von 10/10 erhalten haben

Nächste Schritte

Jetzt, da Sie das Projekt abgeschlossen haben, reichen Sie bitte das Projekt zur Peer-Überprüfung ein. Laden Sie alle Bilder hoch, die Sie beim Erstellen des Projekts aufgenommen haben, wie in der **Einreichung zur Peer-Bewertung** angegeben.

Author(s)

Abhishek Gagneja

© IBM Corporation 2023. Alle Rechte vorbehalten.