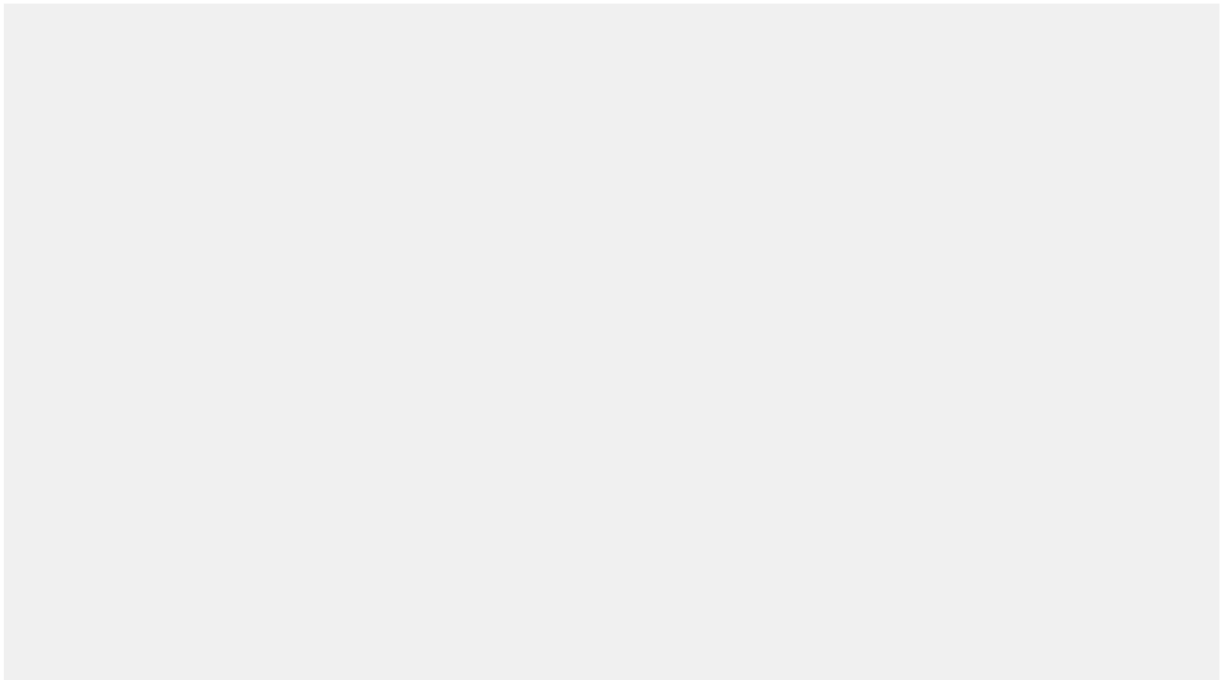





Teil 3



– Aus Antworten lernen –



Viele Nutzer möchten nicht mit einer künstlichen Intelligenz schreiben. Wir richten also eine Verbindung ein, die das Gespräch beendet, wenn der Nutzer sich negativ ausdrückt. Die Verbindung die wir dafür benötigen, könnten wir vielleicht mit Stichwörtern füttern, diese beachten aber nicht den Kontext. Vielleicht ist dir bei der Option *bedingte Verbindung* schon der Button  aufgefallen. Diese wollen wir global einrichten.

NLP – NATÜRLICHE SPRACHVERARBEITUNG (NATURAL LANGUAGE PROCESSING)

Wie ein NLP – Modell entsteht schauen wir uns in Teil 2 an. Jetzt erst einmal die Frage: Warum verwendet man NLP – Modelle? Nehmen wir als Beispiel die Sätze

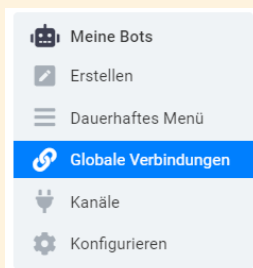
„Ich möchte **nicht** spazieren.“ und „Ich möchte **nicht nur** spazieren.“

In beiden Sätzen steckt das Wort **nicht**. Während im ersten Satz jedoch verneint wird, wird im zweiten Satz (mit Vorbehalt) zugestimmt. Hier ist der **Kontext** der Worte wichtig. Diesen Kontext können sogenannte NLP – Modelle anhand vieler, vieler Beispiele analysieren und dann auf neue Sätze anwenden.

BEISPIEL

Auch hierfür hat [Iris](#) ein Beispiel für dich.

AUFGABE



Erstelle eine Interaktion Verabschiedung.

Wenn der Nutzer sich in einem beliebigen Teil des Gesprächs negativ äußert, soll er direkt zu Verabschiedung geleitet werden. Wir könnten jetzt in jeder Interaktion eine solche Verbindung einrichten, das wäre aber sehr mühselig. Stattdessen erstellen wir eine

globale Verbindung, die immer wirkt. Gehe dazu im linken Menü auf *globale Verbindungen*. Anschließend wählen wir **NLP-Verbindung**

hinzufügen aus. Es öffnet sich ein Menü ähnlich dem von *bedingten Verbindungen*.

The screenshot shows a configuration interface for a chatbot rule. At the top, it says 'Wenn Die Benutzerantwort stimmt mit dem Modell überein.' followed by a dropdown menu set to 'negative words / Entität / (Deutsch)'. Below this, there are three filter sections: 'bei Übereinstimmung mit' (set to 'Entität'), 'danach gehen Sie zu' (set to 'Deutsch'), and 'NLP-Modell' (set to 'Negative Wörter'). There is also an 'Interaktion' dropdown set to 'Verabschiedung' and a 'Webhook' button.

Wähle bei *Typ* **Entität**, bei *Sprache* **Deutsch** und als *NLP-Modell* **Negative Wörter**.

Teste deinen Bot. Bricht er ab, wenn man etwas negatives schreibt? Was eine *Entität* ist und wie diese *NLP-Modelle* funktionieren, erfährst du jetzt.

SCHRITT 2: EIN EIGENES NLP-MODELL

Wir wollen als nächstes ein eigenes NLP-Modell erstellen. Bleiben wir in unserem Beispiel. Ein Tourist hat vielleicht schon mal von einem besonderen Ort gehört, aber kennt den Namen nicht. Trotzdem sollte der Bot natürlich erkennen, welcher Ort gemeint ist. Für unser Beispiel wollen wir ein NLP – Modell erstellen welches erkennt, ob von der [Kunsthofpassage](#) in der Neustadt geredet wird.

NLP – FUNKTIONSWEISE

Wie wird ein NLP – Modell erstellt? Als erstes muss man überlegen, ob innerhalb eines Satzes eine **Entität** oder eine **Absicht** erkannt werden soll.

Eine *Entität* ist ein Objekt, dass einer gewissen Kategorie angehört. Das kann etwas real Existierendes sein, wie zum Beispiel Orte oder Kuchen, aber auch ein Name ist eine Entität. Eine Entität lässt sich innerhalb eines Satzes an einigen Worten erkennen.

Eine *Absicht* hingegen, beschreibt die Intention eines Nutzers. Diese ist natürlich wesentlich schwerer aus einem Satz herauszulesen. Hier ist daher der komplette Satz entscheidend.

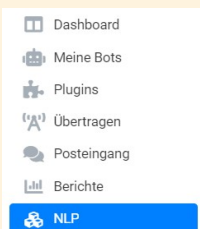
Nachdem man sich entschieden hat, was für eine Art an Modell man trainieren möchte, braucht man Daten. Sehr viele Daten. Diese Daten dienen dem Modell als Referenz: was sind die wesentlichen Eigenschaften auf die ich achten muss? Diese Art des Lernens nennt man **überwachtes Lernen**.

KUNSTHOFPASSAGE

Wir wollen unser eigenes NLP-Modell erstellen. Dazu brauchen wir erst einmal Daten, wie man die Kunsthofpassage beschreiben könnte. Frag doch mal nach den Daten.

Hast du die Tabelle? Super, dann fangen wir mal an.

AUFGABE



Wir öffnen im linken Menü den Reiter **NLP**. Hier findest du das vortrainierte Netz „negative Wörter“, welches wir in Schritt 1 benutzt haben. Unter Benutzerdefinierte Modelle kannst du mit [Ein neues Modell erstellen](#) ein eigenes NLP – Modell erstellen.

Wir wählen als NLP-Typ Entität. Für verschiedene NLP-Typen gibt es unterschiedliche Trainingsansätze. Wir wählen das **Bernoulli-Naive-Bayes-Modell**.

Nachdem das Modell angelegt wurde, muss es mit Daten gefüttert werden. Klicke dazu auf **Proben**. Hier kannst du die einzelnen Sätze aus der Tabelle reinkopieren. Hier ist die Auswahl der Daten entscheidend. Was würde ein Tourist tatsächlich eingeben? Der Kontext welcher Wörter ist entscheidend? Ein Beispiel:

Wo kann man in Dresden Shoppen?

Hier möchte man eher nicht die Kunsthofpassage anbieten, sondern die Prager Straße beispielsweise.

Wo kann man in der Neustadt Shoppen?

Hier bietet es sich wieder an, die Kunsthofpassage anzubieten. Wenn also *shoppen* im Satz vorkommt, sollte dies auch *Neustadt*. Wenn alle Daten zufriedenstellend eingegeben sind, kann mit einem Klick zurück das Modell unter **Trainieren** trainiert werden.

Teste dein Modell und binde es in deinen Bot ein. Erstelle dazu beispielhaft eine Interaktion Kunsthofpassage und füge eine Verbindung von Erklärung zu dieser mithilfe deines NLP-Modells ein.

Funktioniert dein Modell zufriedenstellend? Wenn nicht, dann liegt das wahrscheinlich daran, dass du zu wenige Daten hast. Wie wir uns weitere Daten beschaffen, kannst du im Zusatz erfahren.

ZUSATZ : ANTWORTEN PER WEBHOOK SAMMELN

Moderne KI-Modelle werden mit Millionen von Daten gefüttert. Das ist auch einer der Gründe warum große Tech-Konzerne wie *Google* und *Facebook* führend in der KI – Forschung sind: sie haben einfach viel mehr Daten als alle anderen. Wir wollen für unser kleines Beispiel auch Antworten sammeln.

Wenn die KI in Erklärung nach einem Ort fragt und die Antwort des Nutzers nicht erkennt, richten wir eine Rückfallverbindung ein. Diese führt auf eine Interaktion, die eine Fehlermeldung zurückgibt. Viel wichtiger jedoch ist, dass die Antwort des Nutzers an ein Google-Sheet Document gesendet wird, um sie zu speichern.

AUFGABE

Lege eine neue Interaktion Fehlermeldung an und erstelle eine Rückfallverbindung von Erklärung zu dieser.

Doch wie versenden wir nun die Daten? Dazu nutzen wir einen **Webhook**.

WEBHOOKS

Webhooks sind ein Element des modernen Internets. Dabei verschickt ein Server eine kurze Nachricht an einen anderen Server und informiert, dass ein Ereignis eingetreten ist. Anwendungen werden dann über das Ereignis informiert ohne, dass sie selbst regelmäßig danach **pollen** (nachfragen) müssen.

Richten wir nun das Google Sheets Document ein. Öffne dazu <https://docs.google.com/spreadsheets/>.

GOOGLE KONTO

E-Mail: is63185@outlook.de
Passwort: 1Biber\$ammelt#olz

AUFGABE

Erstelle ein neues Tabellendokument. Öffne nun über **Tools** → **Skripteditor** den Skripteditor. Hier können wir das Dokument so programmieren, dass es Webhook-Daten, die es erhält, in die Tabelle einfügt.

Um mit Webhooks arbeiten zu können, muss das Dokument als Web-App veröffentlicht werden. Web-Apps benötigen zwei Funktionen:

```
function doGet(e) {}           und           function doPost(e) {}
```

Diese stehen für die beiden Anfragemethoden **POST** und **GET**. Da die Web-App beide Methoden empfangen kann, müssen auch beide definiert werden. Da unserer Webhook eine POST-Anfrage sendet, können wir jedoch die GET-Methode sehr leicht wie folgt definieren:

```
function doGet(e) {  
    return HtmlService.createHtmlOutput("doGet request");  
}
```

Als nächstes müssen wir nur noch `doPost(e)` definieren. Innerhalb von `doPost(e)` soll der Request `e` ausgewertet werden.

AUFGABE

Als erstes definieren wir die Registerkarte in die wir schreiben. Diese heißt, sofern nicht geändert „Tabellenblatt1“. Mithilfe von

```
var Registerkarte =  
SpreadsheetApp.getActiveSpreadsheet().getSheetByName(„Tabellenblatt1“);
```

erhalten wir die Registerkarte.

Anschließend müssen wir die gesendeten POST-Daten auswerten. Mithilfe von `e.postData.contents()` erhalten wir den Inhalt der POST-Abfrage.

Anschließend wandeln wir die erhaltenen Daten mithilfe `JSON.parse()` in das **JSON**-Format um, weil wir dieses leicht in das Tabellendokument einspeisen können. Schlussendlich kommt dann auch schon der Befehl mit dem eine neue Reihe an das Tabellendokument angehängt wird:
`Registerkarte.appendRow([daten.message]);`

Noch einmal übersichtlich:

```
var empfangen = e.postData.contents;  
var daten = JSON.parse(empfangen);  
Registerkarte.appendRow([daten.message]);
```

Das ist auch schon der ganze Code. Jetzt müssen wir nur noch unter

Veröffentlichen → Als Web-App einrichten... das Dokument freigeben. Wichtig ist, dass du bei jeder Änderung des Codes als Project Version „Neu“ auswählst.

The screenshot shows the 'Deploy as web app' dialog box. It includes a 'Current web app URL' field with a 'Disable web app' link. Below is a text box containing the URL 'https://script.google.com/macros/s/AKfycbx4Ptn2gOgxC...'. A 'Test web app for your latest code.' button is also present. The 'Project version' is set to '4'. Under 'Execute the app as:', 'Me' is selected. A note states 'You need to authorize the script before distributing the URL.' Under 'Who has access to the app:', 'Anyone, even anonymous' is selected.

Den Webhook kannst du jetzt in deinem Bot testen, indem du ihn an passender Stelle einbindest. Dazu fügst du ihn einfach bei einer Verbindung hinzu. Es wird eine Test-Nachricht an das Google Sheet Dokument gesendet, die du dann dort auch sehen solltest.