

# Spracherkennung, Übersetzung und Textanalyse mit Google Cloud APIs

M.Sc. Onur Yilmaz

Angewandte Künstliche Intelligenz

Schriftliche Ausarbeitung - Cloud Computing  
Fachhochschule Südwestfalen

Gutachter: Prof. Dr. Giefers

13. August 2023

# Inhaltsverzeichnis

<b>1</b>	<b>Cloud Computing und NLP-Technologien</b>	<b>3</b>
1.1	Cloud Computing im Kontext von NLP . . . . .	3
1.2	APIs und ihre Rolle im NLP . . . . .	3
1.3	Interaktion mit der Google Cloud API . . . . .	4
<b>2</b>	<b>Spracherkennung und Transkription</b>	<b>5</b>
2.1	Sprache-zu-Text-Prozess . . . . .	5
2.1.1	Messung und Verbesserung der Sprachgenauigkeit . . .	6
2.2	Cloud Speech API . . . . .	7
2.2.1	Erstellen eines API-Keys . . . . .	7
2.2.2	Erstellen der API-Anfrage . . . . .	9
2.2.3	Aufrufen der API . . . . .	10
<b>3</b>	<b>Sprachübersetzung</b>	<b>14</b>
3.1	Übersetzung von Texten . . . . .	14
3.2	Cloud Translation API . . . . .	16
<b>4</b>	<b>Textanalyse</b>	<b>18</b>
4.1	Klassifizierung von Text in Kategorien . . . . .	18
4.2	Entitäten- und Sentimentanalyse . . . . .	18
4.3	Natural Language API . . . . .	18

# Einleitung

In dieser Arbeit werden Technologien und Anwendungen aus dem Bereich der Sprach- und Textverarbeitung beleuchtet, die auf Google Cloud-Diensten aufbauen. Hierzu gehören die *Cloud Speech API*, die *Cloud Translation API*, die *Natural Language API* und die *Text-to-Speech API* [3].

Im ersten Abschnitt, welches sich auf [7] bezieht, wird eine kurze Einführung in die Konzepte des Cloud Computings sowie der API-Integration im Zusammenhang mit NLP-Technologien gegeben, sowie die Interaktion mit der Google Cloud API erläutert.

Der Abschnitt *Spracherkennung und -transkription* fokussiert sich auf die *Cloud Speech API*, die die Transkription von Audio in Text ermöglicht, und die Methoden zur Messung und Verbesserung der Sprachgenauigkeit.

In der *Sprachübersetzung* wird die *Cloud Translation API* behandelt, die den Prozess der Übersetzung von Texten in verschiedene Sprachen ermöglicht.

Der Bereich *Textanalyse* befasst sich mit der *Natural Language API*, die Techniken zur Klassifizierung von Text in Kategorien und zur Analyse von Entitäten und Sentiments bietet.

Die Arbeit dient nicht nur als theoretischer Überblick, sondern bietet auch praktische Einblicke und Anleitungen zur Verwendung dieser Tools. Dabei werden unterschiedliche Schwierigkeitsgrade und Themenbereiche abgedeckt, um einen umfassenden Einblick in die Möglichkeiten der Sprach- und Textverarbeitung mit Google Cloud zu bieten.

# 1 Cloud Computing und NLP-Technologien

## 1.1 Cloud Computing im Kontext von NLP

Cloud Computing bezeichnet die Bereitstellung von IT-Ressourcen wie Rechenleistung, Speicherplatz und Anwendungen über das Internet. Anstatt Ressourcen physisch vor Ort zu haben, ermöglicht Cloud Computing den Zugriff auf diese Ressourcen in großen Datenzentren. Dies bietet Unternehmen und Einzelpersonen Flexibilität, Skalierbarkeit und oft Kosteneffizienz. Im Kontext der natürlichen Sprachverarbeitung (NLP) revolutioniert Cloud Computing die Verarbeitung, Analyse und Interpretation großer Mengen von Sprachdaten. Es ermöglicht den Zugriff auf leistungsstarke und skalierbare Ressourcen, die NLP-Projekte effizient und effektiv durchführen.

## 1.2 APIs und ihre Rolle im NLP

Wenn man über Cloud spricht, ist eine der Hauptinteraktionen die Verwendung von APIs.

Eine **API** (**Application Programming Interface**) dient als Schnittstelle, die es Entwicklern ermöglicht, bestimmte Funktionen eines Programms oder einer Plattform zu nutzen, ohne sich mit den internen Details auseinanderzusetzen zu müssen.

Das Aufrufen einer API, oft als „API-Anfrage“ (*im engl. request*) bezeichnet und ist der Prozess, bei dem ein Programm oder eine Anwendung eine Anforderung an einen Server sendet und eine Antwort zurückerhält.



Ein Großteil der modernen APIs, insbesondere im Cloud-Bereich, basiert auf dem Prinzip von REST (Representational State Transfer).

**RESTful APIs** nutzen Standard-HTTP-Methoden und bieten einen einheitlichen und vorhersehbaren Mechanismus, um mit dem Server zu interagieren. Dies erleichtert die Integration in Anwendungen und ermöglicht eine breitere Kompatibilität zwischen verschiedenen Systemen.

Im Kontext von NLP und Cloud Computing bieten RESTful APIs eine schnelle, zuverlässige und sichere Möglichkeit, auf leistungsstarke NLP-Modelle und -Dienste zuzugreifen. Sie ermöglichen es Entwicklern, Daten in Echtzeit zu verarbeiten und sofortige Analysen oder Antworten zu erhalten. Zum Beispiel Textdaten zu senden und als Antwort eine Analyse oder Übersetzung dieses Textes zu erhalten, wäre eine beliebte und heutzutage häufig gebrauchte Anwendung ist.



### 1.3 Interaktion mit der Google Cloud API

Die Google Cloud Plattform bietet eine Vielzahl von RESTful APIs, die speziell für NLP-Aufgaben konzipiert sind. Diese APIs profitieren von der leistungsstarken Infrastruktur von Google, welche es Entwicklern ermöglicht, auf fortschrittliche NLP-Funktionen zuzugreifen. Für den Zugriff auf die Google Cloud APIs ist eine Authentifizierung erforderlich, die häufig über **API-Keys** erfolgt. Sobald die Authentifizierung erfolgreich ist, können Entwickler Daten senden, diese verarbeiten lassen und die Ergebnisse für ihre spezifischen Anwendungen oder Analysen abrufen.

## 2 Spracherkennung und Transkription

In diesem Abschnitt behandeln wir den Prozess der Spracherkennung und Texttranskription (siehe [8]), einschließlich der Nutzung der Google Cloud Speech API zur Konvertierung von gesprochener Sprache in Textform [5]. Um hochwertige Transkriptionen zu erhalten, werden wir außerdem auf die Messung und Verbesserung der Genauigkeit bei der Spracherkennung eingehen. Hierbei stützen wir uns auf [4].

### 2.1 Sprache-zu-Text-Prozess

Die Automatische Spracherkennung (ASR), auch als maschinelles Transkribieren oder Sprache-zu-Text bezeichnet, nutzt fortschrittliche Methoden des maschinellen Lernens, um gesprochene Audioinhalte in geschriebenen Text zu verwandeln. Die Einsatzmöglichkeiten sind äußerst vielfältig und reichen von der Erstellung von Untertiteln über die Unterstützung virtueller Assistenten bis hin zur Realisierung interaktiver Sprachsysteme.

Der Prozess beginnt mit der Erfassung der gesprochenen Sprache, sei es mittels eines Mikrofons oder anderer Audioeingaben. Die aufgenommenen Audiosignale werden in diskrete Zeitfenster von typischerweise 10 bis 25 ms unterteilt, die als Sprachrahmen bezeichnet werden. Diese Sprachrahmen fungieren als die elementaren Bausteine, auf denen das ASR-System operiert.

In der nächsten Phase erfolgt die akustische Modellierung. Dieser Schritt umfasst die Analyse der individuellen Sprachrahmen mit dem Ziel, die darin enthaltenen zugrundeliegenden Klänge, als *Phoneme* bekannt, zu identifizieren. Phoneme sind die kleinsten distinkten sprachlichen Einheiten, die den Bausteinen der Wörter zugrunde liegen. Das akustische Modell nutzt ein Trainingskorpus, um auf akkurate Weise die Charakteristiken der Klangmuster in den Audiosignalen zu erfassen und diesen Mustern die entsprechenden Phoneme zuzuordnen.

In der Anschlussphase durchläuft die erkannte Phonemsequenz das Sprachmodell. Das Sprachmodell nutzt die geordnete Abfolge der Phoneme, um kohärente Wörter und Sätze zu generieren, die den erkannten Lauten entsprechen. Dieser Schritt stellt einen essentiellen Bestandteil dar, um die gesprochene Äußerung in einen verständlichen und grammatikalisch korrekten schriftlichen Ausdruck zu transformieren. Am Ende dieses Prozesses stellt das ASR-System den transkribierten Text als Endergebnis dar.

Hier sieht man eine vereinfachte Darstellung dieses Modelles:

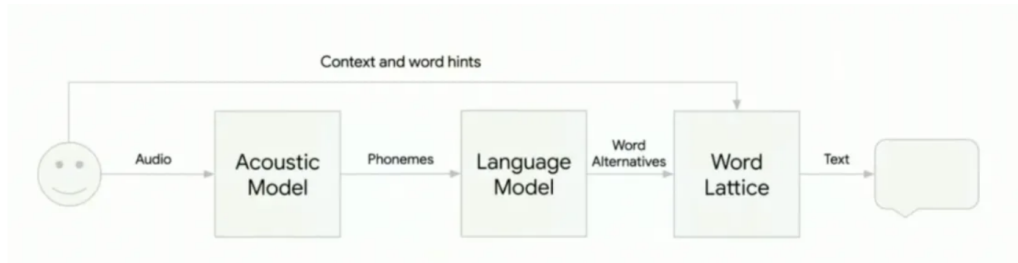


Abbildung 1: *Speech-to-Text* - Prozess

### 2.1.1 Messung und Verbesserung der Sprachgenauigkeit

Die Genauigkeit der Spracherkennung kann auf verschiedene Arten gemessen werden. Je nach Bedarf können mehrere Metriken verwendet werden. Der branchenübliche Vergleich erfolgt jedoch oft mithilfe des Wortfehlerquotienten (Word Error Rate, WER). Die Wortfehlerquote misst den Prozentsatz falscher Transkriptionen im gesamten Satz. Eine niedrigere WER bedeutet daher, dass das System genauer ist.

In Bezug auf die Genauigkeit der ASR sehen Sie möglicherweise auch den Begriff *Ground Truth*. Ground Truth bezieht sich auf die zu 100% genaue (in der Regel menschliche) Transkription, mit der Sie die Genauigkeit vergleichen.

Die Wortfehlerquote setzt sich aus drei Arten von Transkriptionsfehlern zusammen, die auftreten können:

- Einfügefehler (I): Wörter, die in der Transkription vorhanden sind, jedoch nicht in der Ground-Truth-Transkription.
- Substitutionsfehler (S): Wörter, die sowohl in der Transkription als auch in der Ground Truth vorhanden sind, aber nicht korrekt transkribiert wurden.
- Lösungsfehler (D): Wörter, die in der Transkription fehlen, jedoch in der Ground-Truth-Transkription vorhanden sind.

Die Formel für die WER lautet:

$$\text{WER} = \frac{S + D + I}{N}$$

Die Anzahl der Fehler jeder der drei Arten wird summiert und durch die Gesamtzahl der Wörter (N) in der Ground-Truth-Transkription geteilt, um die Wortfehlerquote (WER) zu ermitteln. Dies bedeutet, dass die WER in Situationen mit sehr geringer Genauigkeit größer als 100% sein kann.

## 2.2 Cloud Speech API

### 2.2.1 Erstellen eines API-Keys

Wir beschreiben nun die Schritte, um einen API-Schlüssel zu erstellen und der uns über der SSH bereitgestellten Linux-Instanz verbinden zu können.

Um die Cloud Speech API mithilfe von `curl` anzusprechen, benötigen wir einen API-Schlüssel, der in der Anfrage-URL übergeben wird. Hier sind die Schritte zur Erstellung eines API-Schlüssels:

1. Wir klicken im Navigationsmenü zu „APIs & Services > Anmeldedaten“.
2. Klicken anschließend auf „Zugangsdaten erstellen (*im engl. Credentials*)“ und wählen „API-KEY“ aus.
3. Bewahren die erstellte API-Schlüssel sicher auf, da wir diesen später in unserer Entwicklungsumgebung als Variable verwenden werden.
4. Wir klicken anschließend auf „Schließen“.

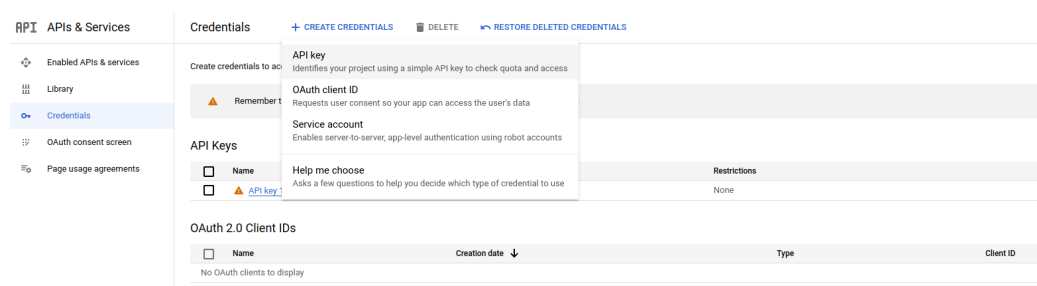


Abbildung 2: Erstellen eines API-KEYs in GCP

Um die nächsten Schritte auszuführen, verbinden wir uns mit der Ihnen bereitgestellten Linux-Instanz über SSH:

1. Navigieren im Navigationsmenü zu „Compute Engine > VM-Instanzen“.



2. Suchen die Linux-Instanz VM in der Liste der VM-Instanzen.
3. Klicken rechts neben dem Namen der Linux-Instanz auf „SSH“, um eine interaktive Shell zu öffnen. Diese Shell verwenden wir, um die nächsten Schritte auszuführen.

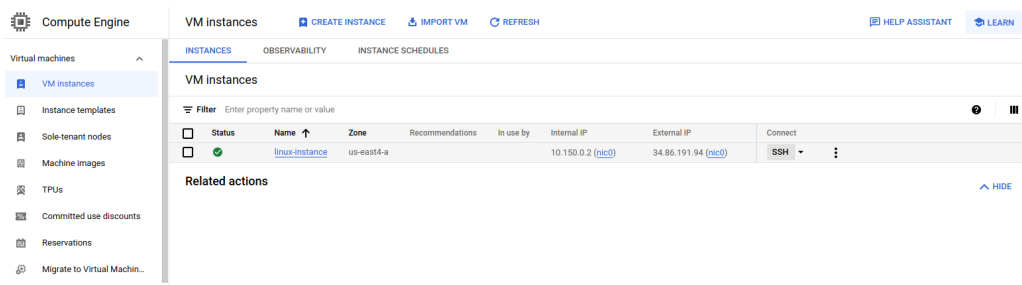
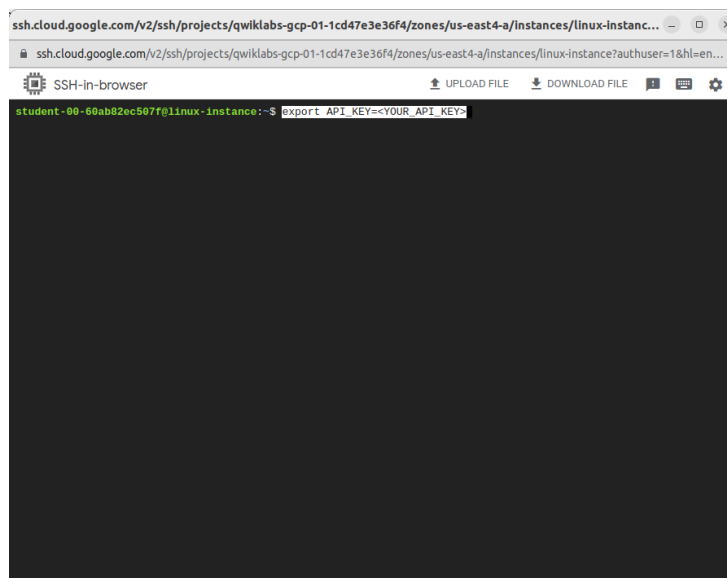


Abbildung 3: VM Instanz

4. In der geöffneten SSH-Shell führen wir den folgenden Befehl aus und ersetzen den KEY durch den von uns zuvor erstellten API-Schlüssel:



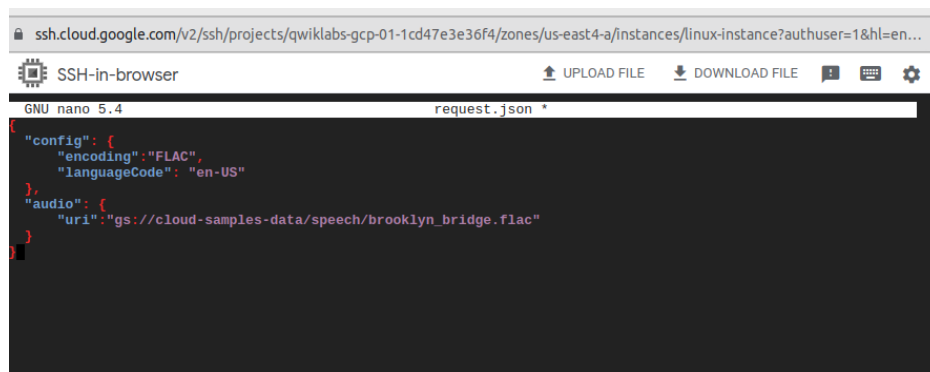
Der nächste Schritt ist die Verwendung von `curl`, um eine Anfrage an die Cloud Speech API zu senden und die gewünschten Transkriptionen zu erhalten.

## 2.2.2 Erstellen der API-Anfrage

1. Wir erstellen unsere Anfrage an die API in einer `request.json`-Datei. Die Datei `request.json` kann wie folgt erstellt werden:

```
touch request.json
```

2. Anschließend öffnen wir die Datei mit unserem bevorzugten Editor (nano, vim, emacs) und fügen Folgendes zu unserer `request.json`-Datei hinzu:



```
ssh.cloud.google.com/v2/ssh/projects/qwiklabs-gcp-01-1cd47e3e36f4/zones/us-east4-a/instances/linux-instance?authuser=1&hl=en...
SSH-in-browser
GNU nano 5.4 request.json
{
  "config": {
    "encoding": "FLAC",
    "languageCode": "en-US"
  },
  "audio": {
    "uri": "gs://cloud-samples-data/speech/brooklyn_bridge.flac"
  }
}
```

Hier obiger Code noch einmal übersichtlich dargestellt:

```
{
  "config": {
    "encoding": "FLAC",
    "languageCode": "en-US"
  },

  "audio": {
    "uri": "gs://cloud-samples-data/speech/brooklyn_bridge.flac"
  }
}
```

Nachdem wir uns den obigen JSON-Code erstellt haben, werfen wir einen Blick über die einzelnen Bestandteile

- „config“: Dies ist ein Objekt, das Konfigurationseinstellungen enthält.
- „encoding“: „FLAC“: Dies legt das Encoding (Datenformat) für den Audiodatenstrom fest. In diesem Fall wird das FLAC-Format verwendet, das ein verlustfreies Audioformat ist.

- „languageCode“: „en-US“: Dies gibt den Sprachcode an. Hier ist es „en-US“, was auf US-Englisch hinweist.
- „audio“: Dies ist ebenfalls ein Objekt, das Informationen über die Audiodaten enthält.
- „uri“: „gs://cloud-samples-data/speech/brooklyn\_bridge.flac“: Dies ist die URI (Uniform Resource Identifier) oder der Pfad zum Audiofile. In diesem Fall handelt es sich um eine Audioaufnahme der Brooklyn Bridge im FLAC-Format, die in der Google Cloud Storage (gs) gespeichert ist.

### 2.2.3 Aufrufen der API

Um die API aufzurufen, müssen Sie den Anfragekörper zusammen mit der API-Schlüssel-Umgebungsvariable an die API übergeben. Dies kann mit dem folgenden `curl`-Befehl erledigt werden (alles in einer einzelnen Befehlszeile):

```
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
student-00-15cf45542a7b@linux-instance:~$ export API_KEY=AIZA6ggMNJmaWaTuYAbTRcCK4pgFLjnOPwKs
student-00-15cf45542a7b@linux-instance:~$ touch request.json
student-00-15cf45542a7b@linux-instance:~$ nano request.json
student-00-15cf45542a7b@linux-instance:~$ curl -s -X POST -H "Content-Type: application/json" --data-binary @request.json \
"https://speech.googleapis.com/v1/speech:recognize?key=${API_KEY}" > result.json
```

Hier noch einmal übersichtlich dargestellt:

```
curl -s -X POST -H "Content-Type: application/json"
--data-binary @request.json \
"https://speech.googleapis.com/v1/speech:recognize?key=${API_KEY}"
> result.json
```

Es handelt sich um einen `curl`-Befehl, der eine POST-Anfrage an die Google Cloud Speech-to-Text API sendet. Die Optionen `-s` und `-X POST` setzen den Befehl auf den silent-Modus und legen die Anfragemethode auf POST fest. Der Header `"Content-Type: application/json"` gibt den Medientyp der Anfrage an. Die Option `--data-binary @request.json` überträgt den Inhalt der Datei `request.json` als Anfragekörper. Die URL enthält den Endpunkt der API, und der Platzhalter `$API_KEY` wird durch den tatsächlichen API-Schlüssel ersetzt. Anschließend wird die Antwort der API in einer Datei

namens `result.json` gespeichert.

Mit dem Befehl `cat result.json` erhält man die folgende Ausgabe:

```
{
  "results": [
    {
      "alternatives": [
        {
          "transcript": "how old is the Brooklyn Bridge",
          "confidence": 0.98216057
        }
      ],
      "resultEndTime": "1.770s",
      "languageCode": "en-us"
    }
  ],
  "totalBilledTime": "2s",
  "requestId": "7372370616388616126"
}
```

Der Wert des „transcript“ gibt die Texttranskription der Audio-Datei durch die Sprach-API wieder, während der „confidence“-Wert angibt, wie sicher die API ist, dass die Audio-Datei korrekt transkribiert wurde.

Um die Google Speech-to-Text API in unserer eigenen Entwicklungsumgebung zu verwenden, ist es notwendig, zuerst das erforderliche Paket über pip herunterzuladen. Anschließend können wir den folgenden Python-Code verwenden, um die API zu nutzen. Dabei übergeben wir die Credentials aus der JSON-Datei und rufen die `recognize`-Funktion auf, um die Spracherkennung durchzuführen: Die Funktion wird auf die selbe FLAC-Audiodatei<sup>1</sup> angewandt.

```
from google.cloud import speech

client = speech.SpeechClient.from_service_account_json('YOUR.json')

config = speech.RecognitionConfig(
    encoding=speech.RecognitionConfig.AudioEncoding.FLAC,
    language_code='en-US',
)
```

---

<sup>1</sup>Auf <https://github.com/ONURYI/Sprache-Text-und-Uebersetzung-mit-GCP-APIs> herunter ladbar

```

with open('speech_brooklyn_bridge.flac', 'rb') as
    audio_file:
content = audio_file.read()

audio = speech.RecognitionAudio(content=content)
response = client.recognize(config=config, audio=audio)

for result in response.results:
transcript = result.alternatives[0].transcript
confidence = result.alternatives[0].confidence

print('transcript: {}'.format(transcript))
print('confidence: {}'.format(confidence))

```

Ausgabe:

```

>> Transkript: how old is the Brooklyn Bridge
      confidence: 0.9821605682373047

```

Zum Vergleich, schauen wir uns nun einmal das von OpenAI bereitgestellte ASR Modell, namens **Whisper AI** (zugehörige Paper siehe [9]) an, welche mithilfe des Paketverwalters pip von Python installiert werden:

```
!pip install git+https://github.com/openai/whisper.git
```

```

import whisper

model = whisper.load_model('base')
model.transcribe('speech_brooklyn_bridge.flac', fp16=False)

```

Und erhalten die folgende Ausgabe

```

{'text': ' How old is the Brooklyn Bridge?',
'segments': [{ 'id': 0,
'seek': 0,
'start': 0.0,
'end': 1.76,
'text': ' How old is the Brooklyn Bridge?',
'tokens': [50364, 1012, 1331, 307, 264, 21872, 18917, 30, 50452],
'temperature': 0.0,
'avg_logprob': -0.3769610643386841,
'compression_ratio': 0.8378378378378378,
'no_speech_prob': 0.019619377329945564}],
'language': 'en'}

```

Hier erkennen wir ebenfalls, dass die Sprache präzise transkribiert wird und auch die korrekte Groß- und Kleinschreibung berücksichtigt wird. Allerdings fehlt uns eine Einschätzung der *Confidence* seitens des Modells.

## 3 Sprachübersetzung

In diesem Abschnitt wird nun die Erkennung und Übersetzung von Texten behandelt (siehe [10]). Hierbei wird ebenfalls auf die Möglichkeiten der *Cloud Translation API* von Google Cloud eingegangen (siehe [6]).

### 3.1 Übersetzung von Texten

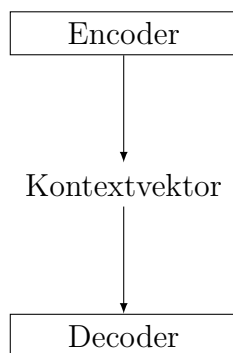
Die Übersetzung von Texten ist eine fundamentale Anwendung im NLP, bei der das Ziel darin besteht, Texte von einer Quellsprache in eine Zielsprache zu übertragen. Ein Schlüsselkonzept, das hierbei eine wichtige Rolle spielt, ist das **Sequence-to-Sequence-Modell**.

Letzteres ist eine weitverbreitete Architektur in der NLP, die speziell für die Übersetzung von Texten entwickelt wurde.

Es besteht aus zwei grundlegenden Teilen:

**Encoder:** Der Encoder nimmt den Eingabetext in der Quellsprache auf, analysiert ihn und erstellt eine kompakte Repräsentation des Textes. Diese Repräsentation, oft als "Kontextvektor" bezeichnet, erfasst die semantischen und syntaktischen Eigenschaften des Eingabetextes.

**Decoder:** Der Decoder verwendet den Kontextvektor des Encoders, um schrittweise den Text in der Zielsprache zu generieren. Während des Generierungsprozesses greift der Decoder auf den Kontextvektor und die bereits erzeugten Teile des Ausgabertextes zurück, um den nächsten Schritt der Übersetzung zu bestimmen.



Wir versuchen das Sequence-to-Sequence-Modell noch einmal mit einem Beispiel zu verdeutlichen.

Angenommen, wir haben die englischen und deutschen Sätze;

Englisch: „How old is the Brooklyn Bridge?“

Deutsch: „Wie alt ist die Brooklyn Bridge?“

Und unser Ziel ist es, vom englischen ins deutsche zu übersetzen. Folgende Schritte werden hier nun durchlaufen:

**1. Tokenisierung:**

Wir teilen die Sätze in Wörter oder Satzzeichen auf:

- Englisch: [„How“, „old“, „is“, „the“, „Brooklyn“, „Bridge“, „?“]
- Deutsch: [„Wie“, „alt“, „ist“, „die“, „Brooklyn“, „Bridge“, „?“]

**2. Vokabular erstellen:**

Wir erstellen Vokabulare für Englisch und Deutsch:

- Englisch Vokabular: [„How“, „old“, „is“, „the“, „Brooklyn“, „Bridge“, „?“, ...]
- Deutsches Vokabular: [„Wie“, „alt“, „ist“, „die“, „Brooklyn“, „Bridge“, „?“, ...]

**3. Token-IDs zuweisen:**

Jedes Wort wird einer numerischen ID<sup>2</sup> zugeordnet:

- Englische Token-IDs: [101, 102, 103, 104, 105, 106, 107]
- Deutsche Token-IDs: [201, 202, 203, 204, 205, 206, 207]

**4. Training:**

Das Modell wird mit den englischen Token-IDs als Eingabe und den deutschen Token-IDs als Ziel trainiert. Der Encoder verarbeitet den englischen Satz und erzeugt einen Kontextvektor.

**5. Übersetzung:**

Der Decoder nutzt diesen Kontextvektor, um den deutschen Satz zu generieren.

Dies ist nur ein sehr stark vereinfachte Darstellung. In der Praxis werden

---

<sup>2</sup>Exemplarisch willkürliche Darstellung.



häufig auf fortschrittliche neuronale Netzwerkarchitekturen wie RNNs, LSTMs und *Attention Layers* zurückgegriffen. Letzteres zeichnet sich durch die Fähigkeit aus, den Kontext eines Satzes zu erfassen.

### 3.2 Cloud Translation API

Die Schritte zur Erstellung eines API-KEYs sind identisch mit dem in 2.2.1 Ablauf beschriebenen Prozess.

Wir wollen hier nun den folgenden Text - „My name is Steve“ ins Spanische übersetzen, hierzu übergeben werden zu übersetzenden Text zusammen mit der API-Schlüssel-Umgebungsvariable an die Übersetzungs-API mithilfe des folgenden Curl-Befehls, direkt in die Cloud Shell ein.

```
TEXT="My%20name%20is%20Steve"
curl "https://translation.googleapis.com/language/translate/
v2?target=es&key=${API_KEY}&q=${TEXT}"
```

```
student_01_c3dd6cd7430c@cloudshell:~ (qwiklabs-gcp-00-9115ce93b694) $ export API_KEY=AIZA5yBQ_Mxj09J1v_h1Hw2AFwaSEL2-cLgchuQ
student_01_c3dd6cd7430c@cloudshell:~ (qwiklabs-gcp-00-9115ce93b694) $ TEXT="My%20name%20is%20Steve"
curl "https://translation.googleapis.com/language/translate/v2?target=es&key=${API_KEY}&q=${TEXT}"
{
  "data": {
    "translations": [
      {
        "translatedText": "Mi nombre es Steve",
        "detectedSourceLanguage": "en"
      }
    ]
  }
}
```

"My%20name%20is%20Steve" ist URL-codiert. Die URL-Codierung ist eine Methode, um Sonderzeichen und Leerzeichen in einer URL darzustellen. Unser Ergebnis lautet dann wie folgt:

```
{
  "data": {
    "translations": [
      {
        "translatedText": "Mi nombre es Steve",
        "detectedSourceLanguage": "en"
      }
    ]
  }
}
```

Zusätzlich zur Übersetzung von Texten kann die Übersetzungs-API auch die Sprache des Textes erkennen. Hierzu geben wir

```
NEW_TEXT="Meu%20nome%20é%20Steven"  
curl "https://translation.googleapis.com/language/translate/  
v2/detect?key=${API_KEY}&q=${NEW_TEXT}"
```

ein und erhalten die folgende Ausgabe:

```
{  
  "data": {  
    "detections": [  
      {  
        "confidence": 1,  
        "language": "pt",  
        "isReliable": false  
      }  
    ]  
  }  
}
```

## 4 Textanalyse

Im letzten Abschnitt liegt der Fokus auf die Textklassifizierung und insbesondere der Entitäts- und Sentimentsanalyse. Hierbei stützen wir uns auf [10]. Wir erörtern außerdem die Anwendungsmöglichkeiten der Natural Language API von Google Cloud, um eine umfassende Analyse von Texten durchzuführen und sie entsprechenden Kategorien zuzuordnen [1], sowie die allgemeine Stimmung zu identifizieren [2].

### 4.1 Klassifizierung von Text in Kategorien

Die Textklassifizierung ermöglicht das Verständnis und die Verarbeitung menschlicher Sprache. Hierbei werden unstrukturierte Textdaten systematisch organisiert und in sinnvolle Kategorien eingeordnet. Dieser Prozess beruht auf der Analyse sprachlicher Merkmale, Schlüsselwörter, Grammatik und semantischer Zusammenhänge im Text.

Ein prägnantes Beispiel für NLP-Klassifizierung ist die automatische Sortierung von E-Mails in Ordner wie „Eingang“, „Werbung“ oder „Wichtig“. Durch die Identifikation von Mustern und Charakteristika im Text erfolgt eine gezielte Zuordnung der E-Mails zu den entsprechenden Kategorien.

### 4.2 Entitäten- und Sentimentanalyse

Die Entitäten- und Sentimentanalyse erweitert die Textanalyse. Entitäten sind spezifische Objekte, wie Personen, Orte oder Organisationen, die im Text erwähnt werden. Die Entitätenanalyse identifiziert und klassifiziert solche Elemente, um eine tiefere strukturierte Darstellung des Textinhalts zu ermöglichen.

Gleichzeitig erfasst die Sentimentanalyse die emotionale Tonalität eines Textes, sei es positiv, neutral oder negativ. Sie basiert auf der Auswertung von Wörtern und Ausdrücken im Text und ermöglicht somit die Einschätzung der übergeordneten Stimmung oder Meinung.

### 4.3 Natural Language API

Die Schritte zur Erstellung des API-Schlüssels sind erneut identisch mit dem in 2.2.1 beschriebenen. Um sicherzustellen, dass diese auch tatsächlich aktiviert sind, können wir in der Suchleiste „*language*“ eingeben und sehen, dass die Aktivierung funktioniert hat.



## Cloud Natural Language API

[Google Enterprise API](#)

Provides natural language understanding technologies, such as sentiment analysis, entity...

MANAGE

TRY THIS API [↗](#)

✓ API Enabled

Mit Hilfe der **classifyText-Methode** der Natural Language API können wir Textdaten mithilfe eines einzigen API-Aufrufs in verschiedene Kategorien einteilen. Diese Methode liefert uns eine Liste von Inhaltskategorien, die auf ein Textdokument zutreffen.

Hierzu navigieren wir wie in 2.2.1 zur Linux Instanz und erstellen folgende Anfrage:

```
{
  "document":{
    "type":"PLAIN_TEXT",
    "content":"A Smoky Lobster Salad With a Tapa Twist.
This spin on the Spanish pulpo a la gallega skips the octopus,
but keeps the sea salt, olive oil, pimentón and boiled potatoes."
  }
}
```

Jetzt könnt ihr diesen Text mithilfe des folgenden curl-Befehls an die classifyText-Methode der Natural Language API senden

```
curl "https://language.googleapis.com/v1/documents:
classifyText?key=${API_KEY}" \ -s -X POST -H "Content-Type:
application/json" --data-binary @request.json > result.json
```

und sehen, dass zwei Kategorien für diesen Text zurückgegeben worden sind:

- /Food & Drink/Cooking & Recipes
- /Food & Drink/Food/Meat & Seafood

```

{ categories:
  [
    {
      name: '/Food & Drink/Cooking & Recipes',
      confidence: 0.85
    },
    {
      name: '/Food & Drink/Food/Meat & Seafood',
      confidence: 0.63
    }
  ]
}

```

Abschließend schauen wir uns nun noch die Entitäts - und Sentimentanalyse mit der Natural Language API an. Die API ist dieselbe es wird lediglich eine andere Methode aufgerufen und zwar die **analyzeEntities-Methode**.

Hierzu wollen wir den Text

"Joanne Rowling, who writes under the pen names J. K. Rowling and Robert Galbraith, is a British novelist and screenwriter who wrote the Harry Potter fantasy series."

in Entitäten aufteilen.

In der Anfrage teilen wir der Natural Language API Informationen über den zu sendenden Text mit. Die unterstützten Typwerte sind PLAIN\_TEXT oder HTML.

```

{
  "document":{
    "type":"PLAIN_TEXT",
    "content":"Joanne Rowling, who writes under the pen names
J. K. Rowling and Robert Galbraith, is a British novelist
and screenwriter who wrote the Harry Potter fantasy series."
  },
  "encodingType":"UTF8"
}

```

Der erste Teil der zugehörige Ausgabe sieht dann wie folgt aus:

```
{
  "entities": [
    {
      "name": "Joanne Rowling",
      "type": "PERSON",
      "metadata": {
        "mid": "/m/042xh",
        "wikipedia_url": "https://en.wikipedia.org/wiki/J._K._Rowling"
      },
      "salience": 0.79828626,
      "mentions": [
        {
          "text": {
            "content": "Joanne Rowling",
            "beginOffset": 0
          },
          "type": "PROPER"
        },
        {
          "text": {
            "content": "Rowling",
            "beginOffset": 53
          },
          "type": "PROPER"
        }
      ]
    }
  ]
}
```

Zu guter letzt können wir mit Hilfe der **analyzeSentiment-Methode** auch die Stimmung des folgenden Textes erfassen:

"Harry Potter is the best book. I think everyone should read it."

Letzteres übergeben wir erneut unserer request Datei.

```
{
  "document": {
    "type": "PLAIN_TEXT",
    "content": "Harry Potter is the best book. I think everyone should read it."
  },
  "encodingType": "UTF8"
}
```

Und erhalten mit dem Befehl

```
curl "https://language.googleapis.com/v1/documents:
analyzeSentiment?key=${API_KEY}" -s -X POST -H "Content-Type:
application/json" --data-binary @request.json
```

folgendes Ergebnis:

```

{
  "documentSentiment": {
    "magnitude": 1.9,
    "score": 0.9
  },
  "language": "en",
  "sentences": [
    {
      "text": {
        "content": "Harry Potter is the best book.",
        "beginOffset": 0
      },
      "sentiment": {
        "magnitude": 0.9,
        "score": 0.9
      }
    },
    {
      "text": {
        "content": "I think everyone should read it.",
        "beginOffset": 31
      },
      "sentiment": {
        "magnitude": 0.9,
        "score": 0.9
      }
    }
  ]
}

```

Es ist zu beachten, dass wir hier zwei Arten von Stimmungswerten erhalten: die Stimmung für das gesamte Dokument und die Stimmung aufgeschlüsselt nach Sätzen. Die Stimmungsmethode gibt zwei Werte zurück:

Score: Ist eine Zahl von -1,0 bis 1,0, die anzeigt, wie positiv oder negativ die Aussage ist.

Größe (Magnitude): Ist eine Zahl im Bereich von 0 bis unendlich, die das Gewicht der Stimmung repräsentiert, unabhängig davon, ob sie positiv oder negativ ist.

## Literatur

- [1] Google Cloud. *Classify Text into Categories with the Natural Language API*. Zugriffsdatum: 08. August 2023. 2023. URL: <https://www.cloudskillsboost.google/focuses/1749?parent=catalog>.
- [2] Google Cloud. *Entity and Sentiment Analysis with the Natural Language API*. Zugriffsdatum: 08. August 2023. 2023. URL: <https://www.cloudskillsboost.google/focuses/1843?parent=catalog>.
- [3] Google Cloud. *Language, Speech, Text, & Translation with Google Cloud APIs*. Zugriffsdatum: 08. August 2023. 2023. URL: <https://www.cloudskillsboost.google/quests/179>.
- [4] Google Cloud. *Measuring and Improving Speech Accuracy*. Zugriffsdatum: 08. August 2023. 2023. URL: <https://www.cloudskillsboost.google/focuses/13597?parent=catalog>.
- [5] Google Cloud. *Speech to Text Transcription with the Cloud Speech API*. Zugriffsdatum: 08. August 2023. 2023. URL: <https://www.cloudskillsboost.google/focuses/2187?parent=catalog>.
- [6] Google Cloud. *Translate Text with the Cloud Translation API*. Zugriffsdatum: 08. August 2023. 2023. URL: <https://www.cloudskillsboost.google/focuses/697?parent=catalog>.
- [7] Prof. Dr. Heiner Giefers. *Cloud Computing*. Skript. 2023.
- [8] Nishit Kamdar. *Measuring and Improving Speech-to-Text Accuracy — Google Cloud Platform*. Zugriffsdatum: 09. August 2023. 2023. URL: <https://medium.com/google-cloud/measuring-and-improving-speech-to-text-accuracy-google-cloud-platform-eba62c50b8ac>.
- [9] Alec Radford u. a. *Robust Speech Recognition via Large-Scale Weak Supervision*. 2022. arXiv: 2212.04356 [eess.AS].
- [10] L. Tunstall, L. von Werra und T. Wolf. *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media, Incorporated, 2022. ISBN: 9781098103248. URL: <https://books.google.de/books?id=7hhyzgEACAAJ>.