Fachbereich
Mathematik, Naturwissenschaften und Informatik

Ausarbeitung

Speed to text

Autor:
Dieutchou, Ruben Chester
Monteagudo Subiria, Joaquin
Ngnindjeu Sonfack, Doriane Lovline

Prüfer: Prof. Dr. Ghobadi.

Abgabedatum: 22.01.2023

II Inhaltsverzeichnis
I Einleitung

Einleitung

Diese Arbeit wurde im Wintersemester 2022 im Rahmen des Moduls "Grundlagen der Künstlichen Intelligenz" erstellt. Ziel ist es, die verwendeten Technologien, Ideen, Lösungen und mögliche Probleme zu dokumentieren.

Wir haben uns für das Projekt "Speech-2-Text" entschieden. Um die Entwicklung zu vereinfachen, haben wir uns dafür entschieden, alle Aufgaben innerhalb einer virtuellen Umgebung mithilfe von Google Collab zu lösen. Die notwendigen Dateien und Anweisungen zur Erstellung der virtuellen Umgebung und zur Installation der Aufgaben finden sich im Github-Projekt. Dort sind die Aufgaben in verschiedene Bereiche unterteilt. Voice Assistants Technologien werden in verschiedenen Anwendungen eingesetzt, wie beispielsweise bei Google Such Engine, Siri, usw.

1 Projektbeschreibung:

Das Projekt beschäftigt sich mit der Anwendung künstlicher Intelligenz zur Konvertierung von Audio Dateien in Text. Dafür haben wir Python als Programmiersprache verwendet und auf Bibliotheken wie Tensor Flow zurückgegriffen. Um das Modell zu trainieren, haben wir ein Datenset von Audiodaten verwendet. Das Modell wandelt die Audiosignale in Frequenzen um, die numerische Werte repräsentieren, die anschließend in Buchstaben umgewandelt werden. Dadurch ist es uns möglich, Audiodateien in Textdateien zu konvertieren. Das Modell wird durch das Training mit dem Datenset verbessert und unser Programm erlernt das Muster der Textübersetzung.

2. Verwandte Arbeit

Ein ARS () hat unterschiedliche Teile die das Problem von Speeech Recognition bearbeiten , wie "Top speech recognition systems rely on sophisticated pipelines composed of multiple algorithms".(Awni Hannun;1;2014)

Normalerweise muss der Modell in 2 Schritte gebaut werden , zuerst wird die Datei für das Modell bearbeitet (pre-processing steps). Dann sollte das Modell Anhand unserer Architektur mit den Daten trainiert und getestet werden .

Bei dem Preprocessing ist öfter ein Acoustic Model und ein Alphabet für die Buchstaben definiert. "Traditional speech systems use many heavily engineered processing stages, including specialized input features, acoustic models, and Hidden Markov Models (HMMs) "

Das Model muss ein bestimmte Art von Lärme bearbeiten , die korrespondieren zur dem Buchstaben einer Sprache. Damit wir besser diese Buchstaben vom anderen Sound unterscheiden, gibt es viele Algorithmen . Diese Algorithmen bearbeiten sowohl die Audiodatei als auch die Output von der Transcription zur verbessern.

Ein Beispiel davon mit der tradionell Architecture gibt es bei der Paper from Hannun and others wo das Model auch ein verbesserung mit eine kombination von den RNN model und das Language Model production für Wörter benutzt ..

"Given the output P(c|x) of our RNN we perform a search to find the sequence of characters c1 , c2 , . . . that is most probable according to both the RNN output and the language model (where the language model interprets the string of characters as words).

Specifically, we aim to find a sequence c that maximizes the combined objective: Q(c) = log(P(c|x)) + α log(Plm (c)) + β word count(c)".(Awni Hannun;4;2014).

In den Quellen wurden 2 Perspektiven zur der ARS modell erfunden.

Einerseits wird der traditional mit algorithmen arbeiten (diese Algorithmen brauche aligned Data). Das bedeutet die werden ein Audiodatei Segment jedes mal analysieren und anhangs das model determinieren, was der Möglichste Wert von den Buchstaben ist und diese in Text umwandeln.

"Traditional HMM (Hidden Markov Models) and GMM (Gaussian Mixture Models) require forced aligned data. Force alignment is the process of taking the text transcription of an audio speech segment and determining where in time particular words occur in the speech segment"(Assemblie 2023:1)

Andererseits kann man deep learning Algorthmen verwenden.

Es gibt viele verschiedene Arten von Architekturen für Deep Learning in der automatischen Spracherkennung. Zwei häufig verwendete Ansätze sind:

1. **Eine Architektur, die auf CNN (Convolutional Neural Network) und RNN (Recurrent Neural Network) basiert und den CTC-Loss-Algorithmus verwendet, um jedes Zeichen der Wörter in der Sprache zu erkennen. Ein Beispiel dafür ist das Deep-Speech-Modell von Baidu.**
2. **Eine Architektur, die auf RNN (Recurrent Neural Network) basiert und jede "Scheibe" des Spektrogramms als Element in einer Sequenz behandelt. Ein Beispiel dafür ist Googles Listen Attend Spell (LAS) Modell.**

Ein besonderer Aspekt einiger dieser Modelle ist, dass sie die Audiodatei direkt(ohne eine language Model) in Text umwandeln können. Dies erfordert die Verwendung von mehreren Algorithmen, von denen die am häufigsten verwendeten die CTC, LAS und RENTs sind.

CTC kann die Probleme von einen eine undeutlichen begrenzung von der EIngabesequenzen vermeiden . Was dies so besonders macht, ist, dass es diese Ausrichtung automatisch durchführt, ohne dass Sie diese Ausrichtung manuell als Teil der beschrifteten Trainingsdaten bereitstellen müssen.

“CTC is used to align the input and output sequences when the input is continuous and the output is discrete, and there are no clear element boundaries that can be used to map the input to the elements of the output sequence.”(towardsdatascience.com :2022)

CTC hat zwei Arbeitsweisen:

Im ersten Modus, CTC Loss, wird ein korrektes Transkript als Ziel verwendet und das Netzwerk trainiert, um die Wahrscheinlichkeit dieses Transkripts zu maximieren.

Im zweiten Modus, CTC Decoding, gibt es kein Ziel-Transkript und das Netzwerk muss die wahrscheinlichste Zeichenfolge vorhersagen.

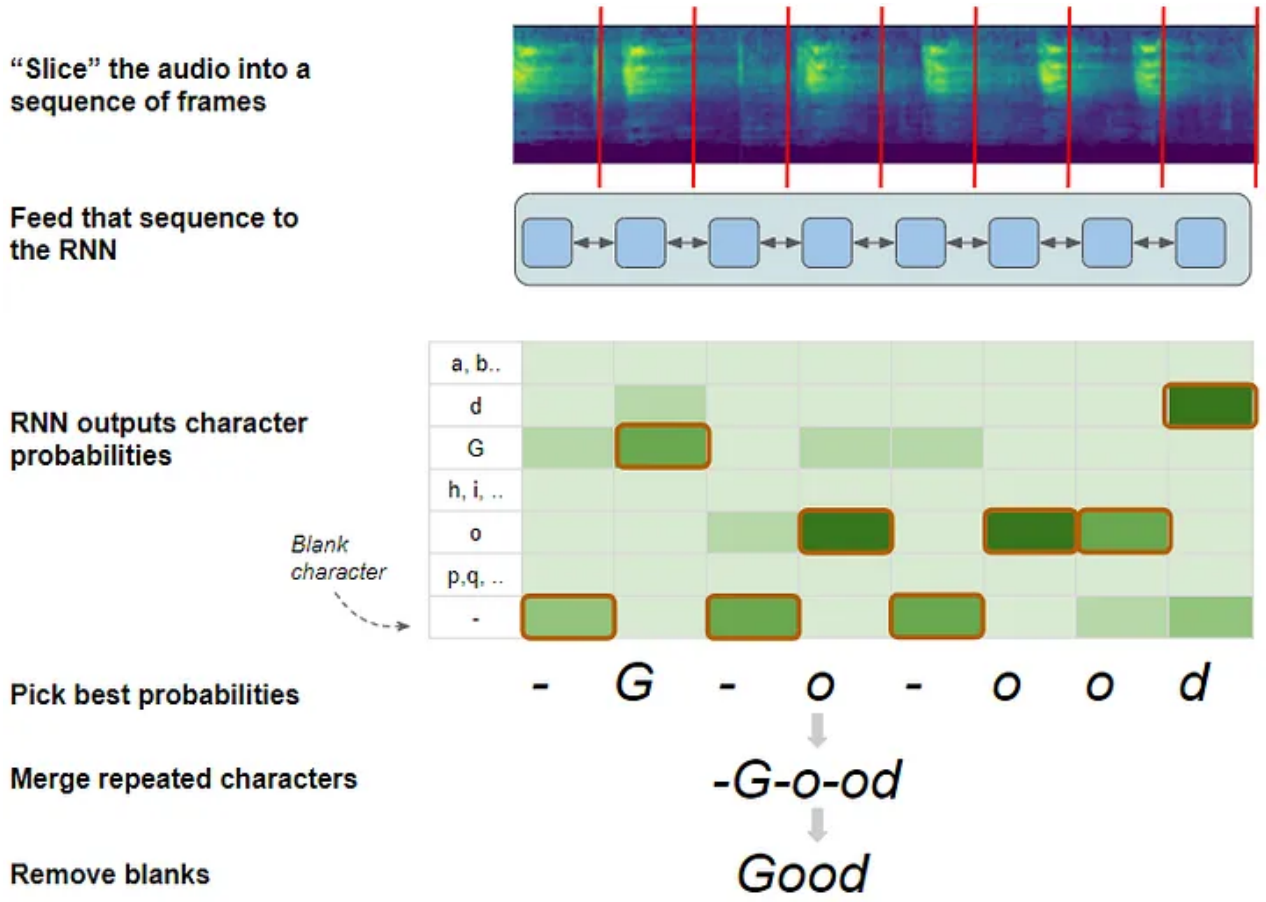
Metriken - Wortfehlerrate (WER)

Eine häufig verwendete Metrik für Speech-to-Text-Probleme ist die Wortfehlerrate (und Zeichenfehlerrate). Sie vergleicht die vorhergesagte Ausgabe und die Zieltranskription, Wort für Wort (oder Zeichen für Zeichen), um die Anzahl der Unterschiede zwischen ihnen zu ermitteln.

EXPLAIN MODEL PREDICTION :

Das Model predicts anhang eine bestimmten spectrogram die mógliche repräsentieren Buchstabe(n) .Die Lange des Spectrogram und das Alphabet wurden vordefiniert . Das Model braucht ein ähnliche Länge für alle Inputdatein , weil jeder Datei den gleichen Format benötigt. Das

“We use spectrograms as (i) our features, so xt,p denotes the power of the p’th frequency bin in the audio frame at time t. The goal of our RNN is to convert an input sequence x into a sequence of character probabilities for the transcription y, with yˆt = P(ct |x), where ct ∈ {a,b,c, . . . , z, space, apostrophe, blank}.”.(Awni Hannun;3;2014)



quelle(https://towardsdatascience.com/audio-deep-learning-made-simple-automatic-speech-recognition-asr-how-it-works-716cfce4c706)

All algorithmen für processing und lernstrategie (KI)

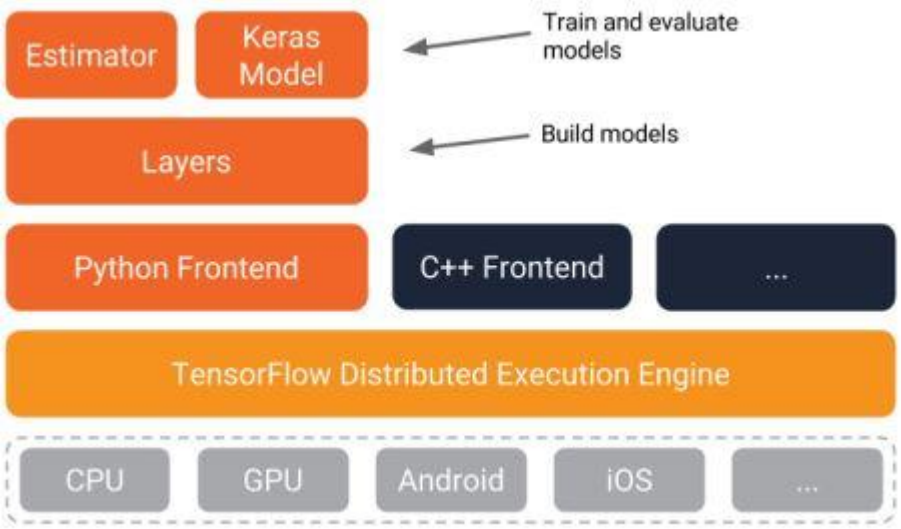
Neuronal netzte..

Audio text model …. deep speech 2 , Google translate audio zur text

1.1 Used Tecnologies: DORIANE

Wichtige Schlüsselbegriffe:

1. **TensorFlow ist eine End-to-End-Open-Source-Plattform für maschinelles Lernen, die von Google Brian entwickelt wurde. Es ist eine symbolische mathematische Bibliothek und wird auch für Anwendungen des maschinellen Lernens wie neuronale Netze verwendet. TensorFlow ermöglicht es Entwicklern, Datenflussgrafen zu erstellen. Dabei handelt es sich um Strukturen, die beschreiben, wie sich Daten durch eine Berechnung bewegen, was es zu einem äußerst flexiblen und leistungsstarken Tool für die Erstellung und den Einsatz von Modellen für maschinelles Lernen macht.**



https://www.tensorflow.org

2. **Neuronale Netze**
   laut https://www.bigdata-insider.de/was-ist-ein-neuronales-netz-a-686185/
   “Künstliche neuronale Netze(KNN) sind inspiriert durch das maschine Gehirn und lassen sich für maschinelles lernen und die KI einsetzen. Es lassen sich mit diesen Netzen verschiedene Problemstellungen computerbasiert lösen. ”

3- Jlwer erklären (Evaluation Model in Bezug auf Fehler von Transkription und Audio

Ein Bewertungsmodell im Kontext von Transkription und Audio bezieht sich auf eine Reihe von Metriken und Verfahren, die zur Bewertung der Leistung eines Transkription-Systems verwendet werden. Das primäre Ziel eines Bewertungsmodells ist es, die Genauigkeit des Transkription-Systems im Hinblick auf die während des Transkription-Prozesses gemachten Fehler zu messen.

Es gibt mehrere Arten von Fehlern, die bei der Transkription auftreten können, darunter:

- **Einfügungen: wenn der Transkription zusätzliche Wörter oder Laute hinzugefügt werden**
- **Löschungen: wenn Wörter oder Laute in der Transkription fehlen**
- **Substitutionen: wenn ein Wort oder ein Laut fälschlicherweise durch ein anderes ersetzt wird**
- **Transpositionen: wenn Wörter oder Laute in der Transkription falsch angeordnet sind**

Um das Transkription-System zu bewerten, wird eine Reihe von Referenztranskriptionen (d. h. die korrekten Transkriptionen) mit der Ausgabe des Systems verglichen. Auf der Grundlage dieser Vergleiche werden dann die Fehler berechnet. Die gebräuchlichsten Metriken zur Bewertung von Transkription-Systemen sind die Wortfehlerrate (WER) und die Zeichenfehlerrate (CER).

Neben der Messung von Transkriptionsfehlern kann ein Bewertungsmodell auch Verfahren zur Bewertung der Qualität der Audioeingabe enthalten. Dies kann die Messung des Signal-Rausch-Verhältnisses, die Überprüfung auf Clipping oder Verzerrung und die Bewertung der allgemeinen Verständlichkeit der Audiodaten umfassen

3-Pandas

Pandas ist eine Open-Source-Bibliothek zur Datenmanipulation und Datenanalyse für die Programmiersprache Python. Sie stellt Datenstrukturen wie Dataframe und Serie zur Verfügung, die für die Handhabung und Manipulation großer und komplexer Datensätze konzipiert sind. Mit Pandas können Sie verschiedene Operationen mit Ihren Daten durchführen, wie z.B. Filterung, Aggregation und Transformation.

4 frequenz, spektrogramm ., code benutz wurde <- NUR DIESEN PUNKT JOAQUIN ….

4 Architecture : DORIANE

Die Erweiterung von Daten durch Spektrogramme beinhaltet die Verwendung von CNN (Convolutional Neural Network) und RNN (Recurrent Neural Network) Architekturen, die den CTC-Loss-Algorithmus verwenden, um jedes Zeichen der Wörter in der Sprache zu erkennen. Beispiele dafür sind das Deep Speech Modell von Baidu und ein RNN-basiertes Sequenz-zu-Sequenz-Netzwerk, das jede "Scheibe" des Spektrogramms als ein Element in einer Sequenz behandelt, wie beispielsweise Googles Listen Attend Spell (LAS) Modell. Es ist wichtig zu beachten, dass unser ultimatives Ziel darin besteht, die einzelnen Zeitschritte oder "Frames" jedem Zeichen im Zieltranskript zuzuordnen.

A convolutional neural network (CNN) is a type of deep learning neural network that is commonly used for image and video recognition tasks. CNNs are designed to process data with a grid-like topology, such as an image, which allows them to learn spatial hierarchies of features from input data. They are composed of layers of interconnected nodes, called neurons, which are organized into multiple layers, including an input layer, one or more hidden layers, and an output layer. Each neuron in a CNN is connected to a small region of the input data, and the network learns a set of weights that can be used to make predictions about the input data.

Connectionist Temporal Classification (CTC) is a loss function used for training recurrent neural networks (RNNs) for tasks such as speech or handwriting recognition. The CTC loss algorithm allows the network to predict a sequence of labels for a given input sequence, even when the alignment between the input and output sequences is not known in advance.

CTC loss is used to train a neural network to predict a sequence of characters or words from a sequence of input features, such as audio features or image features. The network is trained to predict a probability distribution over the set of possible characters or words for each time step in the input sequence.

During training, the CTC loss function compares the predicted probability distribution to the true transcription (label sequence) , and calculates the negative log-likelihood of the true transcription given the predicted probability distribution. The CTC loss function is then used to adjust the weights of the network so that the predicted probability distribution becomes more similar to the true transcription.

CTC loss allows for the use of variable-length input sequences and variable-length output sequences, making it useful for tasks with variable-length input such as speech recognition where the duration of the audio can vary.

It is not possible to write a complete model to predict text from audio data using

TensorFlow in this forum, as it would require a significant amount of code and resources. However, I can provide an overview of the steps that are typically involved in building such a model:

1. **Preprocessing the audio data: The audio data needs to be converted into a format that can be used as input to the neural network. This may involve extracting features such as mel-frequency cepstral coefficients (MFCCs) or spectrograms from the raw audio data.**
2. **Building the model architecture: The model architecture will likely involve a combination of convolutional layers to extract features from the audio data, and recurrent layers such as LSTMs to process the sequential nature of the data.**
3. **Adding CTC Loss function: As the model is going to predict text it will require a CTC Loss function to be added to the model architecture.**
4. **Training the model: The model is trained using the preprocessed audio data and the corresponding transcriptions (text) using the CTC loss function.**
5. **Evaluation: The trained model is evaluated on a held-out set of audio data to measure its performance in transcribing speech to text.**

It is important to note that building such a model is a complex task that requires a lot of data, computational resources and time. It may require more than a simple TensorFlow model and need to use more complex libraries such as Kaldi for feature extraction, etc.

2 Entwicklung von Model :

1. Sammlung von Sprachdaten: Es ist wichtig, eine große Menge an Sprachdaten zu sammeln, die repräsentativ für die angestrebte Anwendung und die Zielgruppe sind. Diese Daten werden verwendet, um das Modell zu trainieren und zu testen.
2. Aufbereitung der Sprachdaten: Die gesammelten Sprachdaten müssen aufbereitet werden, um sie für das Training und die Inferenz des Modells geeignet zu machen. Dies war die Zerlegung der Audioaufnahmen in kleinere Abschnitte und die Anwendung von Signalverarbeitungs Techniken beinhalten.

```python
# An integer scalar Tensor. The window length in samples
frame_length = 256
# An integer scalar Tensor. The number of samples to step.
frame_step = 160
# An integer scalar Tensor. The size of the FFT to apply.
# If not provided, uses the smallest power of 2 enclosing frame_length.
fft_length = 384

def encode_single_sample(wav_file, label):
  ###########################################
  ## Process the Audio
  ###########################################
  # 1. Read wav file
  file = tf.io.read_file(wav_path + wav_file + ".wav")
  # 2. Decode the wav file , decode a 16-bit PCM WAV file to a float tensor
  audio, _ = tf.audio.decode_wav(file)
  # remove 1 dimensional tensor
  audio = tf.squeeze(audio, axis=-1)
  # 3. Change type to float
  audio = tf.cast(audio, tf.float32)
  # 4. Get the spectrogram for Model trainign the fequenz  here for a  frequenz number to a letter)
  spectrogram = tf.signal.stft(audio, frame_length=frame_length,  frame_step=frame_step, fft_length=fft_length)
  # 5. We only need the magnitude, which can be derived by applying tf.abs
  spectrogram = tf.abs(spectrogram)
  spectrogram = tf.math.pow(spectrogram, 0.5)
  # 6. Normalisation
  means = tf.math.reduce_mean(spectrogram, 1, keepdims=True)
  stddevs = tf.math.reduce_std(spectrogram, 1, keepdims=True)
  spectrogram = (spectrogram - means) / (stddevs + 1e-10)
  ###########################################
  ## Process the Label
  ###########################################
  # 7. Convert label to Lower case
  label = tf.strings.lower(label)
  # 8. Split the label
  label = tf.strings.unicode_split(label, input_encoding="UTF-8")
  # 9. Map the characters in label to numbers, sounds , frequenz to number
  label = char_to_num(label)
  # 10. Return a dict as our model is expecting two inputs

  return spectrogram, label
```

3. Auswahl der Modellarchitektur: Es gibt viele verschiedene Arten von Architekturen für Deep Learning, die für ASR verwendet werden können. Wir hatten ein CTC architecture gelernt

We first define the CTC Loss function.

```
def CTCLoss(y_true, y_pred):
  # Compute the training-time loss value
  batch_len = tf.cast(tf.shape(y_true)[0], dtype="int64")
  input_length = tf.cast(tf.shape(y_pred)[1], dtype="int64")
  label_length = tf.cast(tf.shape(y_true)[1], dtype="int64")

  input_length = input_length * tf.ones(shape=(batch_len, 1), dtype="int64")
  label_length = label_length * tf.ones(shape=(batch_len, 1), dtype="int64")

  loss = keras.backend.ctc_batch_cost(y_true, y_pred, input_length, label_length)
  return loss
```

We now define our model. We will define a model similar to DeepSpeech2.

```
[ ] def build_model(input_dim, output_dim, rnn_layers=5, rnn_units=128):
        """Model similar to DeepSpeech2."""
        # Model's input
        input_spectrogram = layers.Input((None, input_dim), name="input")
        # Expand the dimension to use 2D CNN.
        x = layers.Reshape((-1, input_dim, 1), name="expand_dim")(input_spectrogram)
        # Convolution layer 1
        x = layers.Conv2D(
            filters=32,
            kernel_size=[11, 41],
            strides=[2, 2],
            padding="same",
            use_bias=False,
            name="conv_1",
        )(x)
        x = layers.BatchNormalization(name="conv_1_bn")(x)
        x = layers.ReLU(name="conv_1_relu")(x)
        # Convolution layer 2
        x = layers.Conv2D(
            filters=32,
            kernel_size=[11, 21],
            strides=[1, 2],
```

4. Training des Modells: Das Modell wird mit den vorbereiteten Sprachdaten trainiert, um die Fähigkeit zu erlangen, Spracheingaben in Text umzuwandeln.

5. Evaluierung und Optimierung: Das Modell wird evaluiert, um seine Leistung zu messen und die Ergebnisse mit den Erwartungen zu vergleichen. Wenn erforderlich, werden Optimierungsmaßnahmen durchgeführt, um die Leistung zu verbessern.
6. Integrierung und Implementierung: Das ausgebildete Modell wird in die

Anwendung integriert und implementiert, in der

11 Hyperparameter : DORIANE

Hyperparameter sind Einstellungen, die bei der Konfiguration und dem Training eines künstlichen Intelligenz-Modells festgelegt werden, bevor das Modell auf Daten trainiert wird. Sie beeinflussen die Leistung des Modells und können durch Ausprobieren verschiedener Werte optimiert werden, um die bestmögliche Leistung zu erzielen.

Einige Beispiele für Hyperparameter sind:

- Lernrate: Die Lernrate bestimmt, wie schnell das Modell während des Trainings anpasst. Eine höhere Lernrate führt zu schnelleren Anpassungen, aber es besteht auch das Risiko, dass das Modell überfittet. Eine niedrigere Lernrate führt zu langsameren Anpassungen, aber es besteht weniger Risiko, dass das Modell überfittet.
- Anzahl der Neuronen: Die Anzahl der Neuronen in einem Neuronale Netzwerk bestimmt, wie komplex das Modell ist. Eine höhere Anzahl von Neuronen ermöglicht es dem Modell, komplexere Muster zu erkennen, aber es erfordert auch mehr Rechenleistung und kann zu überfitten führen.
- Dropout-Rate: Dropout ist eine Technik, die beim Training von Neuronalen Netzwerken verwendet wird, um das Überfitten zu verhindern. Es bestimmt, wie viele Neuronen während des Trainings deaktiviert werden sollen. Eine höhere Dropout-Rate führt zu einer stärkeren Regulierung, aber es kann auch die Leistung des Modells beeinträchtigen.
- Anzahl der Schichten: Die Anzahl der Schichten in einem Neuronalen Netzwerk bestimmt, wie tief das Modell ist. Eine höhere Anzahl von Schichten ermöglicht es dem Modell, komplexere Muster zu erkennen, aber es erfordert auch mehr Rechenleistung und kann zu Überfittet führen.

12 - Ergebnisse: RUBENS
13 - Metriken: RUBENS
14 - Vergleich mit anderen Modellen (State-of-the-art) : DORIANE & JOA

3 Quellenverzeichnis

[1] Actions. URL: https://rasa.com/docs/rasa/actions/

[2] Baidu Research (2014). Deep Speech: Scaling up end-to-end speech recognition. ArXiv preprint arXiv:1412.5567.

[3] Audio Deep Learning Made Simple: Automatic Speech Recognition (ASR), How it Works (2022) Ketan Doshi. source visited on 21/01/2023 https://towardsdatascience.com/audio-deep-learning-made-simple-automatic-speech-recognition-asr-how-it-works-716cfce4c706

[4] What is ASR? A Comprehensive Overview of Automatic Speech Recognition Technology.Kelsey Foster. Sorce visited on 21/01/2023 in : https://www.assemblyai.com/blog/what-is-asr/

[5]