Karlsruhe Institute of Technology

Code-Switching using Language-Agnostic Speech Modeling

Master's Thesis of

Ugan, Enes Yavuz

at the Department of Informatics
Institute for Anthropomatics and Robotics

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	Prof. Dr.-Ing. Tamim Asfour
Advisor:	M.Sc. Christian Huber
Second advisor:	M.Sc. Juan Hussain

01. June 2021 – 30. November 2021

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, November 30, 2021

. . . . . .

(Ugan, Enes Yavuz)

Abstract

For many years now, research in the area of artificial intelligence has seen a massive surge. Due to numerous developments like increased processing power, increased memory capacities as well as the availability of more data the majority of new developments utilize deep neural networks (DNN). Specifically, in the field of automatic speech recognition (ASR) neural network architectures are utilized to a great extent.
As of today, there are many spoken languages, some with more and some with less available data. Training a separate model for all languages with enough data is possible. Due to restricted capacities on devices that are targeted for deployment, having a separate model for each language is not always possible. Another issue occurs in cases in which the user speaks multiple languages and mixes sentences or even uses different languages during one sentence. These cases are referred to as inter-sentential and intra-sentential code-switching [38]. The issues mentioned above can be addressed by training one model for multiple languages. Such a system can greatly reduce the amount of storage capacity as well as the processing power needed. This however may lead to lower recognition performances as the languages have to share their model parameters.
In order to address and analyse the issues mentioned above, we conduct a series of experiments on multilingual speech recognition in the case of the three languages German, Arabic as well as English. All our models utilize encoder-decoder based sequence-to-sequence systems. The encoder is based on bidirectional Long Short-Term Memory (LSTM) layers [12] and the decoder on unidirectional LSTMs with an additional multi-head-attention mechanism. We will show Word-Error-Rate (WER) results of different models on test sets consisting of monolingual data, and data with intra-and inter-sentential code-switching between aforementioned languages.

Zusammenfassung

Seit einigen Jahren hat die Forschung im Bereich der künstlichen Intelligenz einen starken Aufschwung erlebt. Aufgrund zahlreicher Entwicklungen, wie die verbesserte Rechenleistung, eine erhöhte Speicherkapazität, als auch die Menge vorhandener Daten, fokusuieren sich mittlerweile die Mehrheit der Entwicklungen auf die Verwendung von tiefen neuronalen Netzen. Insbesondere finden diese neuronalen Netze im Gebiet der automatischen Spracherkennung eine sehr intensive Anwendung.
Heutzutage gibt es viele Sprachen die gesprochen werden, dabei haben einige mehr und andere weniger verfügbare Daten. Das Training eines separaten Modells für Sprachen mit genügend Daten ist möglich, aufgrund der kumulierten Größe aller Modelle stellt sich jedoch die Frage der Einsatzfähigkeit solcher Systeme. Da Geräte, welche für den Einsatz vorgeshen sind, begrenzte Kapazitäten habenen ist es nicht immer möglich ein Modell für jede Sprache zu verwenden. Ein weiteres Problem tritt in Fällen auf, in denen der Nutzer mehr sprachig ist. Sätze können dementsprechend in verschieden Sprachen gesprochen werden. Es kann sogar zur Verwendung mehrerer in Sprachen in einem Satz kommen. Diese Fälle werden als inter-sentential und intra-sentential code-switching [38] bezeichnet. Diese angesprochenen Probleme, können durch die Verwendung eines multilingualen Modells angesprochen werden. Dabei werden sowohl die Speicherkapazität als auch die Rechenleistung, erheblich reduziert. Jedoch kann dies zu einer reduzierten Erkennungsleistungen führen, da die Modellparameter unter allen Sprachen geteilt werden müssen.
Um die oben genannten Fragestellungen zu adressieren und zu analysieren, führen wir eine Reihe von Experimenten zur mehrsprachigen Spracherkennung, für die drei Sprachen Deutsch, Arabisch und Englisch durch. Alle unsere Modelle verwenden ein auf Encoder-Decoder basierendes Sequenz-zu-Sequenz System. Der Encoder besteht aus bidirektionalen Long Short-Term Memory (LSTM) Schichten [12] und der Decoder aus unidirektionalen LSTMs, mit einem zusätzlichen multi-head-attention Mechanismus. Wir werden die Ergebnisse der Wortfehlerrate (WER) verschiedener Modelle auf Testsätzen zeigen, die sowohl aus einsprachigen Daten bestehen, als auch auf Datensätzen mit intra- und intersentiellem Codewechsel zwischen den oben genannten Sprachen.

Contents

List of Figures

List of Tables

1. Introduction

Automatic Speech Recognition (ASR) is the task of translating human speech into a sequence of words which can be processed by machines.
ASR systems have many applications in our daily lives as well as in professional work environments. For instance, such systems can be used on mobile devices in order to easily access information by using speech only. Other examples are, starting phone calls, or defining new destinations on navigation systems while driving which helps the user keep their eyes on the street and thus reducing the threat of car accidents. Some example applications in work environments would be Pick-to-Voice system in warehouses or the logging of medical processes during examinations or even operations on patients which reduces the work of the medical staff.
While classic systems modeled the acoustic and the language model separately new approaches utilize neural networks (NN) and model these parts implicitly in one system. Due to the high variation in the data collected, combined with less explicit modeling of phenomenons, these NNs need more training data to yield good performances, when compared to classic approaches. The variation of data is especially apparent in the case of speech recognition. Different dialects, mixing languages, medical conditions which change the voice or multiple speakers speaking simultaneously, which is referred to as the cocktailparty phenomenon, are just a few examples which pose great challenges to current ASR systems. In order to overcome these difficulties we need a large amount of training data that has many representations of these cases. As this is often not the case some fine-tuning for the specific use case needs to be applied to achieve good performance. This however, can lead to the forgetting problem reducing the models capabilities in more general settings. A similar problem of not having enough data to train systems with good performance appears for languages which are not as widely spoken or for which there is not enough data collected. In order to overcome this specific problem, parameter sharing can be applied. In this thesis, one ASR system for multiple languages is trained. This leads to sharing the parameters of one model between all languages, aiming to benefit the language with sparse or less representative training data. This can also help alleviate the problem of recognizing code-switching utterances.
In this Thesis, we will analyse and conduct experiments for multilingual speech recognition in the setting of having no specific language information. We will use sequence-to-sequence models in an encoder-decoder setting based on Long Short-Term Memory (LSTMs) cells as well as a multi-head-attention mechanism which has yielded good results in the past. The model can be seen in subsection 4.1.1. The multilingual setting analysed in this thesis will include three languages:

- Arabic
- German
- English

The baseline models will be trained with all three languages mixed in one training set. We will also analyse how the model performs after training on different multi-task settings in which the second task, next to the main speech recognition one, is defined as predicting the correct language. Models with a custom defined data-set consisting of artificially generated inter-sentential code-switching samples will be trained too. Other models will also include some incorporated language-embedding which will be obtained through the network as well. All experiments will be evaluated with respect to the word error rates (WERs). The results will be analysed on monolingual test sets, as well as on some artificially generated code-switching data.

The following section 2 will briefly describe fundamental pre-processing steps in ASR as well as some basics of neural networks and the architectural components used in our model. Chapter 3 will describe previous works related to this thesis. Work focusing on multi-lingual models as well as aiming at enhancing code-switching performance will be presented. Afterwards in section 4 the model architectures, data-sets and the experimental setup used in this thesis will be described. In the following chapter 5 the performance of the different systems will be analysed on test data. Last but not least in section 6 a conclusion will be given and some interesting future approaches to tackle the tasks of multilingual ASR and code-switching will be discussed.

2. Fundamentals

In this chapter, a short introduction to the topics relevant for this thesis will be given. Subjects covered will be data processing, evaluation metrics, training regimes, and neural networks. For a more sophisticated explanation of the models, we refer the reader to the published papers considering the topic of interest. These will of course be cited in the respective sections.

2.1. Automatic Speech Recognition

Automatic speech recognition describes the task of automatically translating speech into a sequence of words. In the past, automatic speech recognition was divided into many sub-modules such as the acoustic model, the language model, and a pronunciation dictionary. The acoustic model in connection with the pronunciation dictionary was used to model the probabilities of speech with respect to a sequence of words. The language model was used to limit the possible words in the next decoding step as well as getting the best prediction for the next word considering the previous words as well as the probabilities from the acoustic model. The language model was also important in order to differentiate between similar sounding words. These so called classic systems were mathematically described by the fundamental equation.

$$\begin{aligned} \hat{W} &= \arg\max_{W} P(W \mid X) = \arg\max_{W} \frac{P(X \mid W) \cdot P(W)}{P(X)} \\ &= \arg\max_{W} P(X \mid W) \cdot P(W) \end{aligned} \tag{2.1}$$

with W being all possible word sequences and X the pre-processed speech data. $\hat{W}$ represents the target word sequence which maximizes the function. While *P(W)* represents the language model the representation of the acoustic model by $P(X \mid W)$ is not one hundred percent accurate as in the case of speech recognition X represents a continues space and not a discrete one. The mathematically correct way would be using a probability density function instead of a probability distribution. For more information about classic systems please refer to [33].
At first parts of these models where replaced with neural networks. Nowadays fully end-to-end neural network systems have replaced those older sub-models combining all of them in one big neural network. Most of these neural networks utilize an encoder-decoder framework which are statistical models calculating the conditional probability

$$\begin{aligned} P(Y \mid X) &= P(y_1, ..., y_S \mid X) \\ &= P(y_1, \mid X) \cdot ... \cdot P(y_S \mid y_1, ..., y_{S-1}, X)\,, \end{aligned} \tag{2.2}$$

where X denotes a sequence of speech features and Y denotes a sequence of target tokens. In the end the final goal of such ASR systems can be formulated as

$$Y = \arg\max_{Y} \{P(Y \mid X)\}\,, \tag{2.3}$$

which defines the result as the word or token sequence out of all possible sequences, with the highest conditional probability.

2.2. Data Preparation

In this part, the steps for preparing the data in order to train an ASR system, based on a neural network will be explained. In subsection 2.2.1 the pre-processing of the audio data will be described. In the following subsection 2.2.2 an explanation on how to utilize the target text to train a neural network will be given.
Due to its one dimensional nature, one might think that audio signals as a feature should be easier to use for automatic speech recognition, than for example two dimensional images in computer vision. This assumption, however, is quite misleading as there are major obstacles that need to be surpassed and can happen to be overlooked at first sight.
The information given in this chapter is based on [33]. The main difficulties of speech signals can be described as continuity, variability, complexity, and ambiguity. Two of which can be tackled in the pre-processing step. The problem of continuity becomes clear when plotting a speech signal. In such signals it is generally hard to find the beginning and the end of graphemes, syllables, or even words. The second major difficulty is the very high variability in recorded speech signals. This variance is caused by many factors, such as the hardware used for the recording, the structural surrounding of the microphone, the person who is speaking, the mood of the speaker, or even health conditions, just to name a few. All of these parameters affect the raw speech signal to a great extent and as such make the use of it a non optimal approach. This will be further explained in the audio pre-processing step, subsection 2.2.1. Example speech signals of the same utterance are given in Figure 2.1. Another obstacle for successful automatic speech recognition is the complexity of the task which not only needs big storage capabilities in order to store the audio data but also has a very huge search space. A sentence with the length of L and a language with K words has K^L possible word combinations. Reducing the number of words is addressed during the text preparation. Last but not least the ambiguity of spoken words, so called homophones, words that have the same pronunciation but different meanings is another difficult task to solve in the context of speech recognition. An example of such words would be the ”bank” on which people can sit and the ”bank” in which money is kept.

2.2.1. Audio Pre-Processing

In order to be processed by commonly used digital computers, the analog signal needs to be converted to discrete values. This is done by sampling the signal. Research has shown that information in the speech signal above 8 kHz is redundant. This is why the signal is low pass filtered to only allow for frequencies below 8kHz and as such a sampling rate of 16kHz is sufficient to avoid aliasing effects on such low pass filtered signals. This sampling rate is also used for all the experiments in this thesis. Another point is to quantify the values of the signal, which is usually done with 16bit, as it was shown that only 12bit is needed to quantify the data with an information loss which can not be distinguished by human perception. Afterwards in order to do a short-time spectral analyses a Hamming window on a frame of 25 ms is applied.

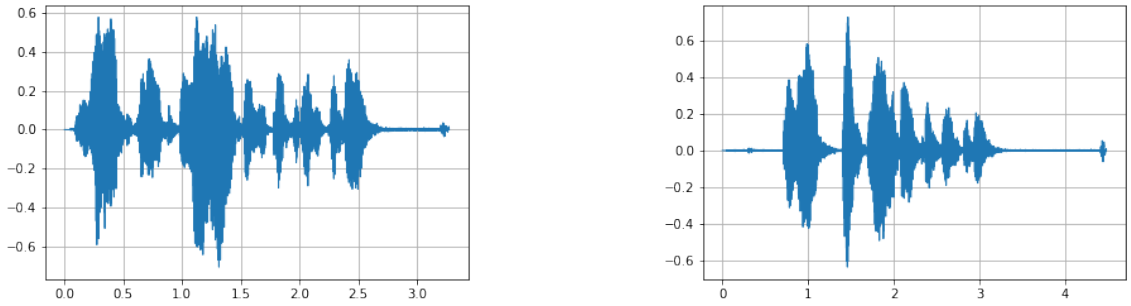

Figure 2.1.: Two signals of the same utterance ”rob ate my biscuits yesterday” recorded by two different speakers. The x-axis depicts the time dimension and the y-axis the signal amplitude.

These windows need to be small enough for the signal, to be considered stationary. As we apply a Discrete Fourier Transformation on these windows, the trade off between frequency resolution and time resolution becomes visible. Bigger windows yield a higher frequency resolution but low time resolution and smaller windows vice versa. This is why the window size was set to 25 ms in this thesis which has shown to be a good size in previous research. In order to improve the resolution in time and account for changes at border regions of the window, the frames have an overlap of 15 ms. In these windowed signals the Fast Fourier Transformation Algorithm is calculated. As the input are real values the output after taking the absolute of the discrete Fourier Transformation are symmetric discrete values. This is why the data size can be reduced in half. Afterwards the spectrum is Mel-scaled and the logarithmic values are taken as features. In the past a common step was to apply an inverse Fourier Transformation and use the resulting Mel-Cepstrum's as features. In this thesis, however, we stopped after taking the logarithm of the Mel-Spectrum's. Figure 2.2 shows an example output of an utterance after the pre-proccessing step. The y-axis depicts the logarithmic Mel coefficients which we set to 40 and the x-axis shows the number of frames. Afterwards this is fed into the model as the input feature.

2.2.2. Text Preparation

Another important step to train automatic speech recognition systems is the text data preparation, as well as the definition of what the labels of the system should be. We applied the following pre-processing steps. Numbers were replaced with their corresponding word representation. All brackets and punctuation marks were deleted. Additionally, the text is also lower-cased, as it was previously shown to yield better performance than mixed-cased text [10].
Another important decision is what kind of target labels should be used. A few possible options are using phonemes, graphemes, Byte pair encodings [9], among many others. In this work, we decided to leverage the Byte pair encodings. Byte pair encoding tries to encode a given text with a specific amount of tokens. Tokens can be words, parts of words, or even only letters. In our case, we generated 4000 tokens as target labels. To achieve representative tokens in

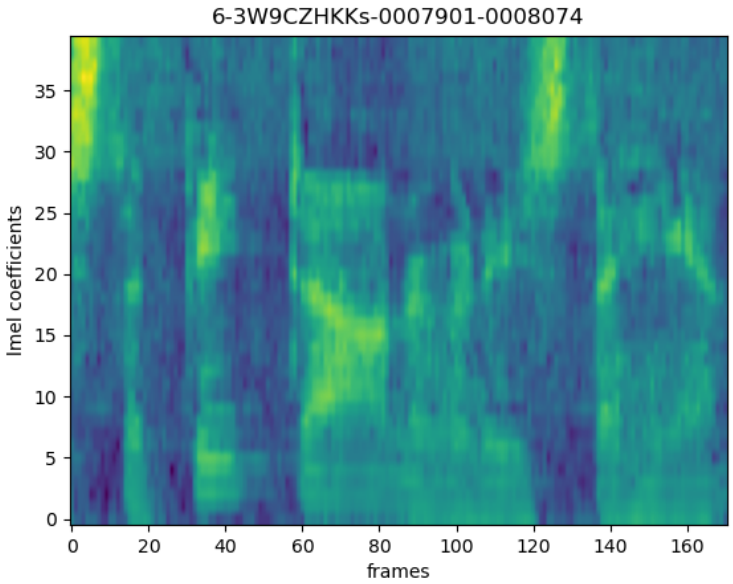

Figure 2.2.: Logarithmic Mel filter bank coefficients for the utterance ”the feet down and release the hands”, with the identifier: 6-3W9CZHKKs-0007901-0008074. Brighter colors depict higher values.

our multilingual setting, we mixed together text sets of the three languages German, Arabic, along with English and applied the Byte pair encoding (BPE) algorithm on it. However there are a few points which need to be considered here. First of all the different amount of available text data for the languages. If one language is more prominent in the text, the BPE result will be heavily favoring that language in the resulting tokens, meaning the tokens will be able to represent that language very well but the other languages will suffer heavily. Another point arises in the Arabic text corpus which has an overwhelming amount of news data and only small amounts of general data from different domains. If used as such, the BPE tokens will be focused on encoding text related to news but not general texts which in return will lead to a harder recognition task for the resulting ASR system on general domain data. Following above mentioned circumstances we first generate an Arabic text file with an over sampled amount of general non news text. We than shuffle all our text files containing the three languages separately. Afterwards we only pick the first 100.000 lines of each file and mix them into one file. This resulting text data is than used to calculate the BPE tokens, which are used as target labels for the multilingual systems used in this thesis.

2.3. Neural Networks

In this section, a brief introduction of the most general neural network architectures used in this thesis will be given.

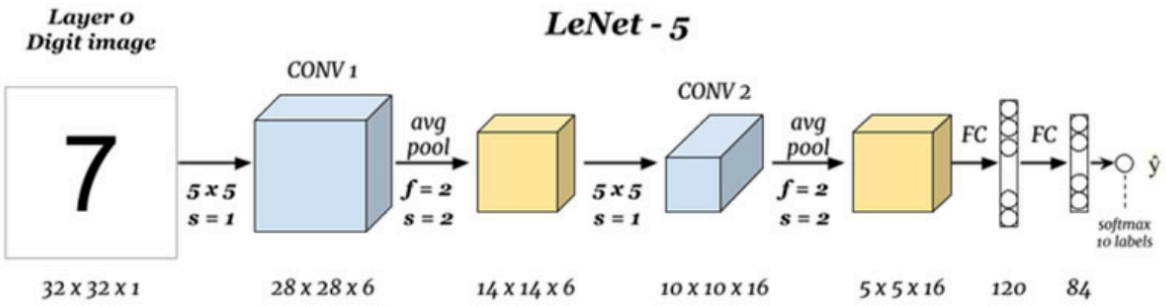

Figure 2.3.: LeNet-5 architecture [19].

2.3.1. Convolutional Neural Network

In order to achieve invariance towards same features occurring at different times in a speech signal at first, a Time Delay Neural Network (TDNN) was proposed by Waibel et al. in 1987 [45]. This yielded significant improvements over classic systems in the task of phoneme recognition. TDNNs can also be considered a form of one-dimensional convolutional networks.
Similar to TDNNs developed in speech recognition, researchers in the field of computer vision developed a two-dimensional convolutional network, generally only called convolutional neural network (CNN). These CNNs aim at increasing invariance in two dimensions as well as reducing the number of parameters when compared to fully connected layers. Nowadays CNNs are commonly used in automatic speech recognition systems as well.
Convolutional neural networks combine architectural designs such as local receptive fields, weight sharing, and spatial and temporal sub-sampling which to some extent achieve the benefit of shift, scale and distortion invariance [18]. The kernel size determines the so-called local receptive area and has one weight for each element. Determined by the stride this kernel is then moved over the input and can be used for sup-sampling. While moving over the capture, the weights of the kernel are kept the same, for each element, which is where the weight sharing takes place. Using multiple such kernels resembles learning multiple feature extractors which aim at finding different features all across the input.
Some of the most prominent models successfully utilizing the CNN architecture are the AlexNet [17], VGG-16 [40] as well as the LeNet-5 [18] also shown in Figure 2.3.

2.3.2. Recurrent Neural Network

In the case of speech recognition, acoustically but also semantically, what is said during the speech at one point has some dependency towards what was said at some previous point in time. If we consider phonemes, for example, different languages have more often occurring transitions between some phonemes in comparison to other transitions, which may even not occur at all. Considering the grammar and vocabulary of languages it is save to say that there are graphemes and word transitions which are more common than others and some are never seen. Due to these settings in which dependencies need to be put into consideration, a specific network, to model these dependencies over time was developed, the recurrent neural network (RNN). One might think that CNNs are also able to capture some time dependencies if one of the convolutions is done over the time dimension. However, due to the need of having

features over multiple time-steps in advance, this needs a more complex data pre-processing. In contrast, RNN only have one input at a time and are still able to capture dependencies over multiple steps.
In the paper, [48] it is shown, that in the task of natural language processing (NLP) CNNs perform better if the task is a classification task which can be solved considering some key words or key parts of the input. RNN-architectures, however, outperform CNNs if there are longer dependencies in an input sequence in which the beginning has a significant effect on the output. They also show that classification based on short independent parts, as in CNNs, may mislead the model's results. In the ASR setting, its clear that utilizing RNN structures can be quite beneficial.

Probably the most basic RNN is the Elman Network. In Figure 2.4 an extracted and simplified example of the proposed network in [8] is shown. In this system, the previous hidden state $h_{(t-1)}$ is copied and used together with the input from the current time step $x_{(t)}$ to calculate the new hidden units $h_{(t)}$. The combination of the current input with the previous hidden state are determined by trainable parameters. Following, it is possible to say that the output $y_{(t)}$ at step t is now influenced by previous inputs. A recurrent network if unfolded can also be pictured as a feed forward network without cycles, as shown in Figure 2.5. Here it is possible to observe that all inputs are transformed using the same weights W^I. The output and the transition "between" hidden states respectively share their weights W^O, W^H as well. Looking at the unfolded structure in the image one can more easily comprehend that in order to learn the weights of a recurrent model it is needed to back-propagate the error through time as $h_{(t-1)}$ influences all hidden units and as such the output values after time step t. In Figure 2.5 if we wanted to calculated the gradient for $h_{(t-1)}$ and the sequence ended at $t+1$ the calculations would look like

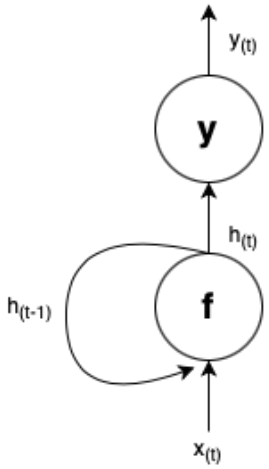

Figure 2.4.: Simplified Elman-Network architecture.

$$\frac{\delta L}{\delta h_{(t-1)}} = \frac{\delta L_{(t+1)}}{\delta h_{(t+1)}} \frac{\delta h_{(t+1)}}{\delta h_{(t)}} \frac{\delta h_{(t)}}{\delta h_{(t-1)}} + \frac{\delta L_{(t)}}{\delta h_{(t)}} \frac{\delta h_{(t)}}{h_{(t-1)}} + \frac{\delta L_{(t-1)}}{\delta h_{(t-1)}}, \tag{2.4}$$

where $L_{(t)}$ and $h_{(t)}$ denote the loss calculated, and the hidden state, at time t. Looking at equation 2.4 one potential problem becomes visible. In the left part of the addition, we can see a chain of multiple multiplications which may lead to either exploding or vanishing gradients, depending on the weights being slightly higher or lower than one. One possible solution for this kind of problems are Long Short-Term Memory cells, which will be discussed in subsection 2.3.3.

2.3.3. Long Short-Term Memory

As mentioned in the previous subsection 2.3.2 the Long Short-Term Memory (LSTM) [12] is a building blog that can be used to implement a recurrent system which deals with the problems of vanishing and exploding gradients in applications with longer input sequences. As the authors in [12] mention, the constant error back-propagation over many time steps

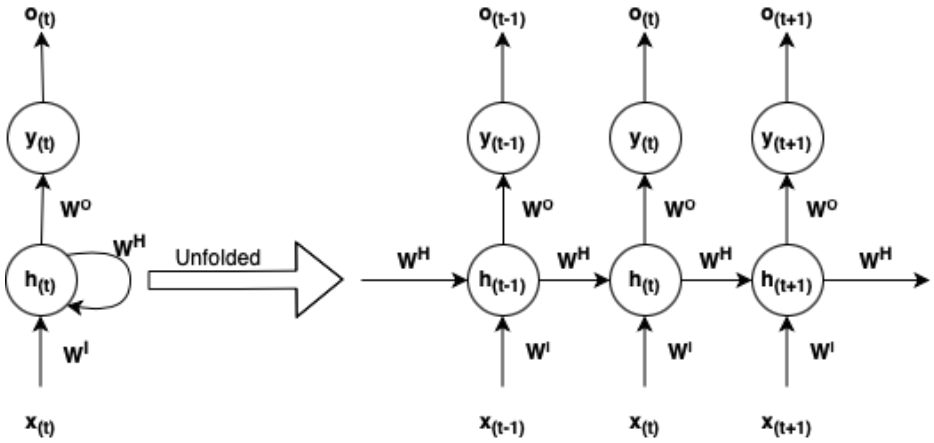

Figure 2.5.: Unfolded RNN in time axis.

enables the models to learn longer dependencies over the time dimension as well. An example visualization of an LSTM is given in Figure 2.6. Here $x_t \in \mathbb{R}^{d_{model}}$ depicts an input vector at time *t*. $h_{t-1} \in \mathbb{R}^{d_{model}}$ represents the hidden state of the the previous LSTM-cell. In case of the first input usually a zero-vector is used. The vector $c_{t-1} \in \mathbb{R}^{d_{model}}$ illustrates the cell state vector which plays a major role in learning long term dependencies. The W values denote weight matrices. There are four important components of an LSTM cell. The three gates.

- The forget gate:
$$f_t = \sigma(W_f x_t + b_f + W_{hf} h_{t-1} + b_{hf}) \tag{2.5}$$
Intuitively said, this gate regulates how much information from the previous cell state should be kept for the current one. This can also be analysed mathematically as the Sigmoid activation function only outputs values between zero and one in which, zero would lead to a loss of information, and 1 would completely retain the information.

- The input gate:
$$i_t = \sigma(W_i, x_t + b_i W_{hi} h_{t-1} + b_{hi}) \tag{2.6}$$
The input gate determines how much information of the current content will be added to the cell state. As it is shown the activation is chosen to be the Sigmoid function resulting in a similar behaviour as the forget gate. The current context is given as the linear combination of the current input x_t with the previous hidden state h_{t-1}. The element-wise multiplication is then able to adjust the influence of each dimension in the linearly transformed current context g_t which is afterwards accordingly added to the long term cell state c.
$$g_t = \tanh(W_g x_t + b_g + W_{hg} h_{t-1} + b_{hg}) \tag{2.7}$$

- The output gate:
$$o_t = \sigma(W_o x_t + b_o + W_{ho} h_{t-1} + b_{ho}) \tag{2.8}$$
Here the current content is used to decide which information of the cell state should be outputted, using a Sigmoid function.

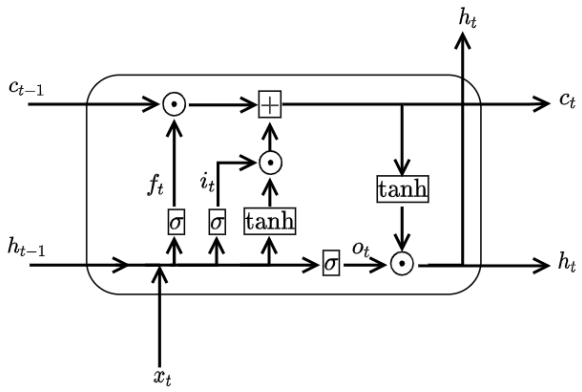

Figure 2.6.: Detailed display of an LSTM-cell and its connections.

The fourth important part of the LSTM-cell is the cell state which is influenced by the above mentioned gates.

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2.9}$$

Looking at Figure 2.6 and Equation 2.9, it is possible to see that there are multiple additive errors which are now back-propagated from the previous time steps. These can balance each other out and thus prevent the gradient from vanishing or exploding. Another important point is the direct influence of the cell state at time t to the next step $t + 1$. This enables the error to be back-propagated in a way which is more direct without being influenced by any weight matrices and specifically enables the system to have a more meaningful gradient over longer sequences. To give a more clear view of how cell states are connected, Figure 2.7 shows an example of how three sequential inputs x are processed in an LSTM-cell.

2.3.4. Bidirectional LSTM

The idea of the bidirectional LSTM (Bi-LSTM) is the potential loss of the surrounding information when creating the output at time t as the hidden state is only based on the current and previous inputs. In order to also account for the information which is following in the next steps, the bidirectional LSTM was proposed. Basically, a second LSTM is used and fed the information the other way round, with respect to the time axis. The output of both LSTMs is than combined into one output at each step t. This combination can be done in varying ways, per addition or concatenation just to name a few.

2.3.5. Attention

Previous encoder-decoder networks were designed to encode the input into a fixed-length hidden representation and the decoder would decode that vector. The authors in [4] suggest that such a fixed-length representation is a bottleneck for these models. And as such, they propose to use an attention mechanism which decides for itself which parts of a variable-length encoder output are relevant to predict the new output. Another effect of the attention module is the potential alignment of the input with the output. In speech recognition this would be an alignment of speech frames with output tokens which is other wise a non-trivial task, as there

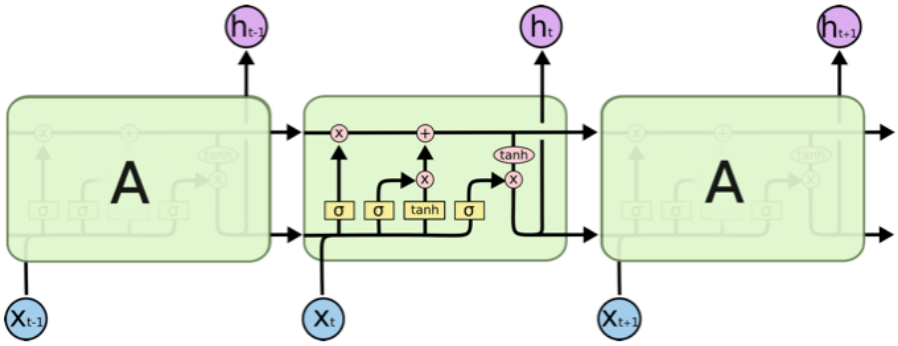

Figure 2.7.: LSTM-cell with three sequential inputs.

are generally more speech frames than output tokens. In the following a description of the attention mechanism implemented in this thesis will be provided.
Given $h_j \in \mathbb{R}^{d_{model}}$, as the outputted hidden vector of the encoder for position $j \in \mathbb{N}$ with feature dimensions d_{model}, the encoder output can be written as $H \in \mathbb{R}^{J \times d_{model}}$. The output from the decoder token embedding is $S \in \mathbb{R}^{I \times d_{model}}$, where I depicts the number of inputted tokens in the decoder. $s_i \in \mathbb{R}^{d_{model}}$, is accordingly the embedding of the i-th output token, .
Now S and H are transformed to so called queries, keys and values via linear transformations.

$$q_i = W_Q s_i \tag{2.10}$$

$$k_j = W_K h_j \tag{2.11}$$

$$v_j = W_V h_j \tag{2.12}$$

Each of these linear transformation are calculated using a separate weight matrices $W_Q, W_K, W_V \in \mathbb{R}^{d_{model} \times d_{model}}$. Next up a similarity score is calculated using the dot product of the query $Q \in \mathbb{R}^{I \times d_{model}}$ with the transposed Key matrix $K^T \in \mathbb{R}^{d_{model} \times J}$.

$$eij = \frac{q_i k_j^T}{\sqrt{d_{model}}}, \tag{2.13}$$

the result being $e \in \mathbb{R}^{I \times J}$. In order to avoid results with too high values causing unstable gradients during training, a scaling factor is introduced to the similarity metric. Afterwards, for each row of e, a Softmax is calculated over the similarity values, as depicted in Equation 2.14. This results in a normalized vector defining the importance of each encoder output for the next hidden state of the decoder with respect to the previous state which is encoded in the query.

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{J} e_{ik}} \tag{2.14}$$

Last but not least for each row of α the weighted sum over the values $V \in \mathbb{R}^{J \times d_{model}}$ is calculated.

$$c_i = \sum_{j=1}^{K} \alpha_{ij} v_j \tag{2.15}$$

The result is also called the context vector $c_i \in \mathbb{R}^{d_{model}}$. In this project, this is implemented via matrix multiplication and yields the matrix $C \in \mathbb{R}^{I \times d_{model}}$.
A special case of the above mentioned attention module is called self-attention. In this case, all input parameters q, k, and v are of the same origin and thus the name "self".

2.3.6. Multi-Head-Attention

In subsection 2.3.5 the basic attention mechanism was described. One possible limitation of this system is that there might be multiple encoder frames containing different but nonetheless important information. In order to predict a token at time t, encoder frames not just around frame t but also at future steps further ahead might be important. However, having only one module could only invest so much on specific parts of the input as due to the nature of the Softmax distribution all probabilities add up to one. In order to enhance the representational prowess of such modules the multi-head-attention (MHA) was developed.
After the first linear transformation of the query, key, and value inputs, those vectors are split depending on the number of heads. In our thesis, this number is defined as eight. This means a query now looks like this:

$$Q \in \mathbb{R}^{i \times 8 \times (d_{model}/8)} \tag{2.16}$$

Generally speaking the Equations 2.13, 2.14 and 2.15 are now calculated for each head separately on queries, keys and values as depicted for queries in Equation 2.16. At the end of the attention after calculating the context from Equation 2.15 for all heads, the context vectors of the different heads are concatenated back to one vector with the original dimension d_{model}. An example image of the attention result of a model used in this thesis is given in Figure 2.8. Shown here are the attention matrices for the eight attention heads for one example utterance. Brighter colors depict higher influence of the corresponding encoder output at the respected position of the output sequence. In all heads the alignment effect between acoustic feature and the target label sequence, mentioned in subsection 2.3.5, are visible. In some cases the heads are more focused on specific encoder frames for example in head five. In others like head six the attention is more divergent and focuses interestingly on the last encoder frame. As one would expect, a diagonal importance trace can be seen which roughly shows which position in the acoustic corresponds to which output token.

2.3.7. Dropout

In the context of artificial intelligence and specifically neural networks, dropout describes the probability with which a neuron is disabled during one training iteration. In our thesis a dropout of 0,2 was applied on specific layers, this means that about 20% of the neurons in these layers are disabled during training. One idea behind the dropout is to enable the model for better generalization and preventing feature co-adaptation. Another important point, however, is the theory that each dropout can be considered to train a separate model. These models have shared parameters as the still active neurons and their connections are still the same. During inference the dropout is not applied, which results in a collective model of many systems and thus yields better performance. In [11] the authors show how a random dropout of neurons prevents the models from over-fitting as well as increases its performance. Although their

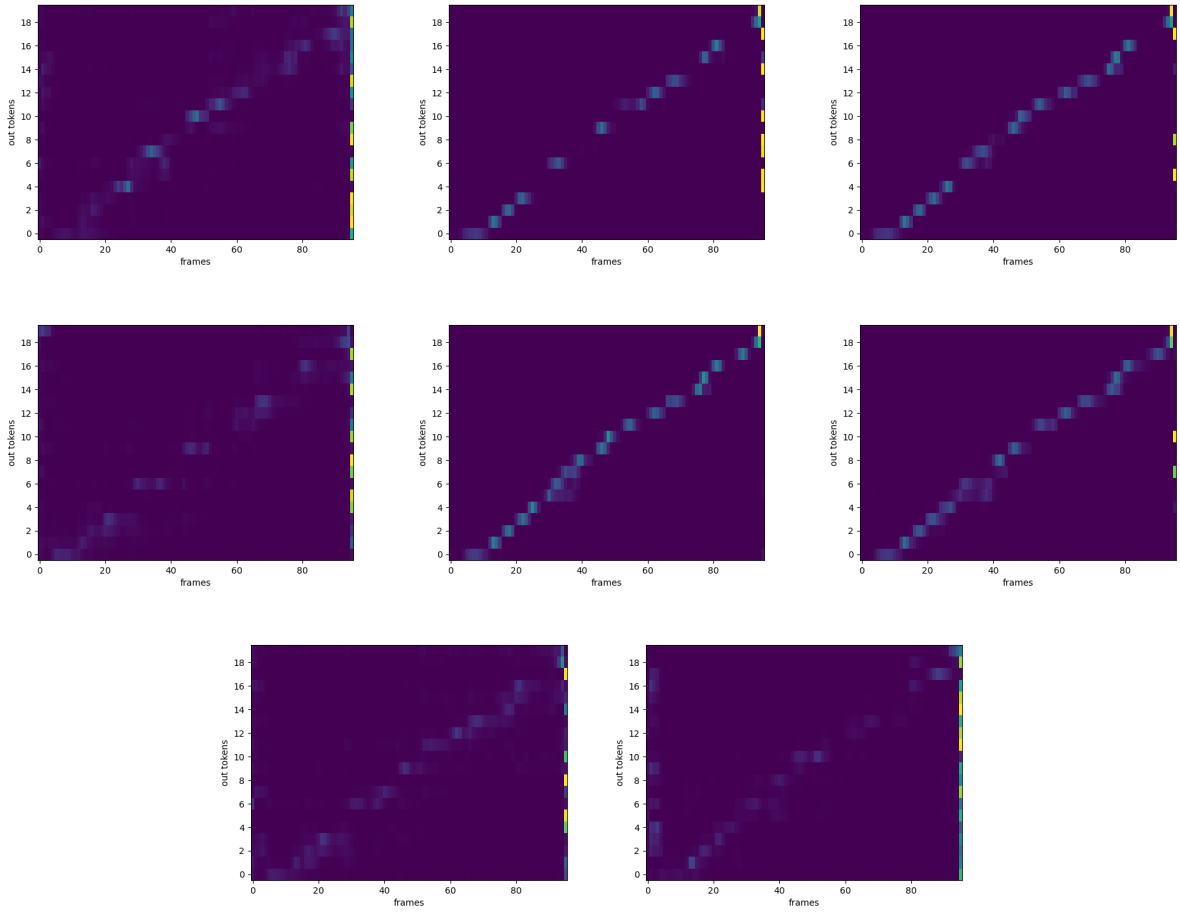

Figure 2.8.: Eight attention heads of the Seq2seq model described in subsection 4.1.1. From top left to to bottom right are the heads one to eight. The utterance is: ”haben sie sich beinahe täglich wertlos oder schuldig gefühlt”.

example shows results in the task of computer vision, the effects persist in other tasks such as ASR or neual machine translation as well.

2.3.8. Encoder-Decoder Networks

Encoder-decoder Networks are made up of two parts. The first part is the encoder and the second one is the Decoder. The encoder network can be made up of any kind of layers or sub-modules, for example, the ones mentioned in section 2.3. The idea is to encode an input into a hidden representation which is usually not human interpretable. The decoder is made up of different layers and sub-modules similar to the encoder and tries to decode the information given in the hidden representation. In [43] the authors talk about the problem normal deep neural networks have with variable length inputs and outputs and propose a sequence-to-sequence model based on an encoder-decoder introduced in [7]. They use this setup in the context of translation and map the variable length input sentence in one language to a fixed sized vector. This vector is then used in the decoding steps as an extra input and as such

provides the system with some compressed information about the text which was inputted to be translated. This example shows a general use case of an encoder-decoder framework in neural machine translation as well as in tasks such as automatic speech recognition.

2.4. Evaluation Metrics

In this section, two ways of evaluating a speech recognition system are explained. While the perplexity (subsection 2.4.2) is used to evaluate the model during training, the Word error rate (subsection 2.4.1) is used as a final evaluation of the model to have some comparable numbers with systems of other labs or researchers.

2.4.1. Word Error Rate

In the field of speech recognition one of the most commonly used evaluation metrics is the word error rate (WER) or its aberrations like Character Error Rate (CER) in the case of languages which do not use an alphabet as the writing system, like Chinese or Japanese.
The WER is a metric comparing the hypothesis prediction of an ASR model with its ground truth reference text. The value of the metric is defined by the minimum number of insertions #ins, substitutions #sub and deletions #del which are needed to transfer the hypothesis transcription into the reference text. The sum of those values is then divided by the number of words #words in the reference text and than multiplied with 100%. Accordingly a number closer to zero percent is desirable. In the case of CER the same is done as in WER only with characters instead of words. The calculation is shown in Equation 2.17.

$$WER = \frac{\#ins + \#sub + \#del}{\#words} * 100\% \tag{2.17}$$

This WER is usually calculated on a separate test set data which was not seen during training.

2.4.2. Perplexity

In the field of Information theory, the perplexity (ppl) can be described as a metric of how uncertain a probability model is when it comes to predicting a specific output. This means that high values show that the model has a hard time predicting the expected sample, while a low value means that there is little confusion and the outputted probability distribution of the model is similar to the expected one.
In order to determine the improvements of the training process, in this thesis, the perplexity of the model on some cross-validation data is used. This metric is used to decide whether the model has trained to saturation and as such an early stopping can be applied, and to restrict the training time and prevent the model from over-fitting on training data. The calculation is shown in Equation 2.18.

$$ppl = \frac{1}{N} * \exp(\sum_{n=1}^{N} \ln(q(x_i))) \tag{2.18}$$

N denotes the total number of words which are to be predicted. $q()$ is the probability distribution of the ASR model and as such, $q(x_i)$ depicts the probability of the model predicting the correct token x at time i.

2.5. Curriculum learning

Inspired by the structured way of how humans learn, starting from simple tasks during childhood towards complex problems in the following years, the authors in [5] propose a training schema called curriculum learning. The main idea is to utilize previously learned concepts which are easier, in order to simplify the learning of more complex tasks for a DNN system. They also suggest that curriculum learning might be comparable to the unsupervised pre-training of a model, helping the system to converge faster towards a better solution. How currciulum learning is applied in this work will be explained in subsection 4.3.2.

2.6. Sequence-to-sequence Models

Sequence-to-sequence (seq2seq) Models are artificial neural networks working with sequences of inputs. As the name suggests the output of such a system is also a sequence. These models can be used for many different kinds of tasks. An example use case would be describing a short video clip. Here, the input would be a sequence of pictures and the output would be some text describing what was shown in the video. Other examples are Neural Machine Translation (NMT) which essentially is treated like seq2seq task in state of the art approaches [29]. Here the input is a variable length sequence of words in language A and the target should be the corresponding translation in a language B. Important to note is that usually not only the input sentences but also the output sentences are made up of a variable number of words which makes it hard to use non seq2seq models, as they often times do not provide necessary flexibility. Another task which has rather similar circumstances when it comes to the variability of input and output lengths is automatic speech recognition. Here the input sequence are the features calculated over the frames of the original speech signal. The output sequence are the tokens or other possible classification tasks, which can be used to decode towards text, as previously mentioned in subsection 2.2.2.

2.7. Code-switching

Code-switching (CS) in speech is referred to as the act of changing back and forth between languages or varieties of one language while speaking [22]. There can be multiple reasons for a person to use more than one language while speaking. If a person is learning a new language and struggles explaining his thoughts, this person might start speaking his native language in order to be able to communicate more easily. Another motive could be expressing solidarity, as speaking or using words of another language can establish a rapport with the addressed listener as well [41]. It could also be done for convenience reasons. In bilingual communities it can be used to communicate faster with each other without anybody missing out on information. Code-switching can be divided into multiple categories [30]:

- Inter-sentential CS: The switch between languages happens at sentence boundaries. Usually, the speaker is aware of the language shift.
- Intra-sentential CS: Here the second language is included in the middle of the sentence. This switch mainly occurs unaware of the speaker. Additionally, the word used from the second language can happen to be adapted to the grammar of the major language as well.
- Extra-sentential CS: In this case, a tag element from a second language is included, for example at the end of a sentence. This word is more excluded from the main language.

Out of the three CS examples presented here, this thesis focuses only on the first two categories.

3. Related Work

This chapter will briefly cover previous work related to this thesis. As there are two tasks which are looked into, research considering the two areas of language-agnostic multilingual speech recognition and systems enhancing the performance in a code-switching setting will be described.

3.1. Language-agnostic multilingual seq2seq models

In this section, an overview of previous approaches on language-agnostic multilingual ASR systems will be given.
There is a long tradition in investigating multilingual systems and using more diverse data to enhance the capabilities of classic automatic speech recognition systems, mentioned in section 2.1. In [42] the authors investigated the multilingual training of articulatory feature detectors. They show that pooling feature detectors from multiple languages outperforms monolingual ones. They also demonstrate that their ASR performance improved when incorporating feature detectors trained on multiple languages. In another work [34], the authors present that by using language independent acoustic models for cross-language transfer, the recognition accuracy for low resourced languages can be improved significantly. The authors of [35] also show results improving their systems capabilities when adapting a multilingual system towards a new language instead of only training a monolingual system. These observations suggest that there is great potential in training language-agnostic ASR models in general.
In contrast to classic models, nowadays, similar to the model used in this thesis, the overwhelming majority of automatic speech recognition systems are based on all neural network models. This being said there are only few approaches considering fully end-to-end trainable neural network based language-agnostic systems.

3.1.1. Mixing training set of multiple languages

One basic approach of training multilingual models can surely be mixing the data sets of multiple languages into one set. It could also be considered to use some specific mixing ratio when generating the training set or even in a later stage considering the ratio of the batches seen during training. The first of these two approaches was proposed in [25]. In this paper three data sets of Russian, Kazakh and English are combined into one. The amount of data, however, is very unbalanced. Specifically the availability of transcribed English audio is available in larger amounts when compared to Russian or Kazakh data. This is why the writers suggest to sample a similar amount of audio duration for all languages in order to train their model. Target labels used in there work are graphemes. As the considered languages have different grapheme sets they try two different approaches of combining them. The first method is to

utilize the adjunction of the three grapheme sets.

$$G_{all} = G_{kz} \cup G_{ru} \cup G_{en}$$

Here the G represents the grapheme set for the respective language. In the second method, the authors concatenate the three sets by adding language identifiers to each grapheme.

$$G_{all} = G_{kz} + G_{ru} + G_{en}$$

As there are shared graphemes between Russian and Kazakh the second approach slightly increases the number of target tokens.
The authors then compared monolingual models of each language with multilingual ones using the target tokens as described earlier. The results show that the multilingual system has comparable results to the monolingual ones. Interesting to note is that the WER on the English test set with Kazakh accent was reduced by up to 3,3% from 41,6% WER for the monolingual English model to 38,3% WER in the multilingual setting.

3.1.2. Prediction language identifier and text output

In [46] the authors propose a monolithic multilingual system for ten languages. In order to achieve language independence, the union of graphemes appearing in all languages namely, English, German, Spanish, French, Italian, Dutch, Portuguese, Russian, Japanese, and Chinese is taken as target labels. They also add language identifiers (LIDs) to the target set. When a target sequence is predicted, the model is trained to first predict the language ID and then the rest of the sequence of labels. An example of a German prediction would have the form

"[DE] ALSO VORWÄRTS" ,

taken from [46]. In contrast to [25] presented in subsection 3.1.2, this work does not sample the languages to have a similar share in the training data.
The authors show that especially in cases for low-resourced languages the CER is reduced by 2-3% on average compared to their monolingual counterpart models. Important to note is that the authors incorporate an additionally learned language model, which is learned separately. This makes the approach not a fully end-to-end system. Leaving this part out, however, would make it end-to-end trainable. A similar approach to this one was also used to improve the CER in the code-switching task. In that case instead of predicting the language label one time at the beginning, the language needs to be predicted every time the language is changed in the utterance. This will be further elaborated in subsection 3.2.1.

3.1.3. Using language feature vectors

In [24] the writers utilize a learned language feature vector (LFV) to enhance the performance of their model. In contrast to the first two approaches subsection 3.1.1 and subsection 3.1.2 this system is not trained in an end-to-end way. In the first step some model for extracting bottleneck features from the input audio feature is trained. The next step uses these bottleneck features and trains a DNN which classifies nine languages. The second last layer of the language

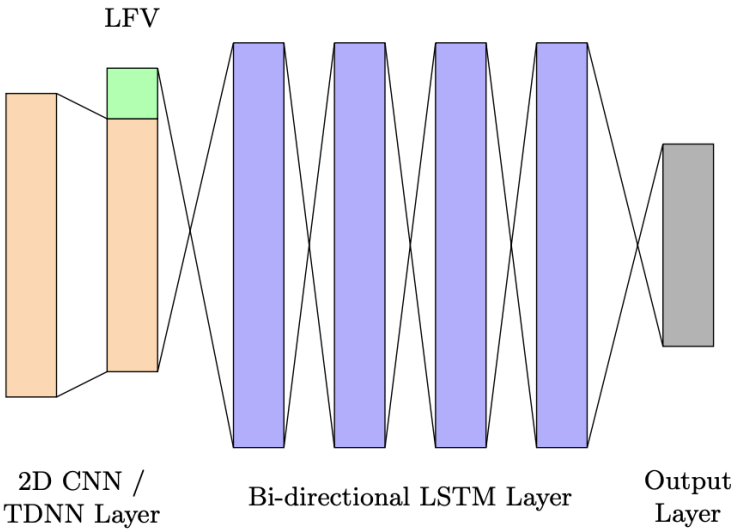

Figure 3.1.: Multilingual Network utilizing LFV's.

classification network is than used as the language feature vector. After putting the speech features through two convolutional layers the outputs are than concatenated with the LFV for that speech sequence and inputted into a multi-layer neural network. This network is than trained separately utilizing the Connectionist Temporal Classification (CTC) loss. Figure 3.1 shows the model architecture and how the LFV is utilized.
In their experiments, the incorporation of the LFV increases the performance of the multilingual models and can even surpass monolingual models if trained with an early abortion of the training on mono-lingual cross-validation set.

3.2. Code-switching in seq2seq models

In this chapter an overview of previous research on the topic of recognizing speech that contains code-switching will be given. In ASR, code-switching is referred to as the act of switching between two or more languages during an utterance. This switch can happen in multiple ways.As described in section 2.7, if the language changes at sentence boundaries it is called inter-sentential, if the switch between languages occurs inside one sentence it is referred to as intra-sentential CS. As there is only very few data available, there has been only sparse research on this topic, most of them utilizing one of the few code-switching data sets that are available. Some example corpora available are [2] for code-switching between French and Algerian speech, [21] containing utterances switching between Mandarin and English and [6] having gathered data with CS between English and Cantonese.

3.2.1. Predicting language switch

Similar to subsection 3.1.2 the authors of [36] propose a model which has the union of graphemes of all languages plus language specific tags as the target label. However, in order to have better results in the task of code-switching, they suggest artificially generating training data that contains code-switching utterances. In order to achieve this, they combine full length utterances

of different languages. A maximum number of speech sequences to combine is defined. First one utterance is drawn, the next time two utterances are drawn and the third time three sequences are concatenated, this is repeated until the audio duration of the union of the original corpora is reached. The probability of sampling from a language is proportional to the duration of that language in the unified corpus. In order to prevent the same speech from occurring too often in the training set they also define a number limiting how many times a utterance is allowed to be used for generating CS data. When concatenating the corresponding targets the language specific token is also added before the target sequence of the respective utterance. An example target sequence would than look like:

"[DE] ALSO VORWÄRTS [EN] WHERE ARE YOU"

In this paper, it is also proposed to use curriculum learning similar to section 2.5. In the first step, the authors train the multilingual models on data without code-switching. In the second step, they take the model from step one and train it with above mentioned code-switching data. In their results, there is up to 26,35% CER improvement comparing curriculum training with directly training on code-switched data. Interesting to note is that the authors register CER improvements in almost all languages comparing the multi-lingual setting with the monolingual one, even if they have big amounts of data. Their multilingual system even outperforms the monolingual models if no CS data was used during training, which might be due the increased amount of parameters in their multilingual network.
In [49] it is also proposed to use a language prediction before predicting target labels of the next language. Aside from using a different model architecture, the authors were also able to utilize the SEAME data-set [21] which contains code-switching data between Mandarin and English. As they only looked into the task of code-switching between those two languages they only use the that data and report their results on a test set containing CS between Mandarin and English speech with South East Asian accent.
They show that including the prediction of the language tag yields improvements of up to 1,2 mixed error rate (MER) which is a combination of WER and CER. Adjusting the output posteriors by the previous language id additionally improves their model.

3.2.2. Multi-task learning

Another approach which was able to utilize data containing natural code-switching was presented by the authors of [39]. However, they propose to solve the problem of code-switching using a multi-task learning (MTL) approach. The authors investigate training a model predicting a sequence of labels as well as predicting a language ID at different levels. The effect, of classifying the language is measured on the following three layers of the network.

1. The context vector which is outputted by an Attention layer
2. The context vector after the residual connection from the decoder
3. The output of the decoder which contains information about the language model

A visualization of the proposed model is given in Figure 3.2 which was taken from [39].

Without the multi-task setting the writers report a CER of 8,15%, the best improvement was achieved using the language classification on the context of the attention. This yielded a CER of 7,60%. They also reported even further improvements when they used monolingual speech data as well and fine-tuned the model with code-switching data afterwards. This resulted in their best performing model achieving 6,49% CER. However, the authors are mainly concerned with the performance on code-switching, and as such do not report the performance of the model on monolingual data which may have degraded and perform worse than a monolingual model.
This issue was addressed in [37]. Here the authors show that indeed the performance degrades on monolingual test sets if only code-switching data is used. They show that using monolingual and code-switching data during training yields better results. An amount of 25% of data containing code-switching resulted in the best performing model. Alternatively, they also suggest using a regularization when fine-tuning on code-switched data. Here the Kullback-Leibler divergence between a previously trained monolingual models output distributions with the model which is fine-tuned for CS, is minimized. The combination of the CTC loss with this additional Kullback-Leibler regularization also improved on the basic models which combined monolingual and code-switching data during training.

3.2.3. Code-switching with frame level language identifiers

In [20] the authors propose to train two separate models. One CTC model for speech recognition and another one for frame level language prediction. During decoding, if the current frame has a very high probability for the blank symbol the blank label is emitted, otherwise the output probabilities of English tokens are multiplied with the probability of this frame being English and the Chinese labels are multiplied with the probability of this frame being Chinese speech. An overview of proposed architecture, which was taken from [20] is given in Figure 3.3. The authors mention that CTC models are very sensitive to initialization and as such they suggest to train the model first on monolingual data of the majorly used language in their code-switching set. Afterwards they train their model a second time utilizing all data. Two different approaches are proposed for the frame based language classification. One is based on Bi-LSTMs and the other one is a DNN with a context window of 41 frames. In their experiments, they show that the highest accuracies for frame-level LID prediction are achieved when the models are trained only on code-switching data. Another important point is that the Bi-LSTM based model outperforms the DNN model if code-switching data is used during training but performs worse in the case of not seeing code-switching data, which seems to be expected due to the memorizing nature of the LSTM

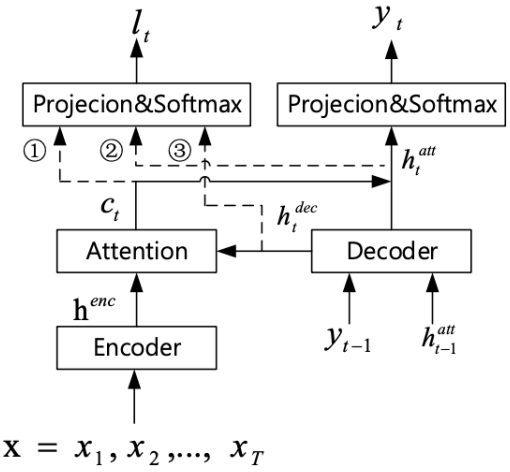

Figure 3.2.: Encoder-Decoder with MTL.

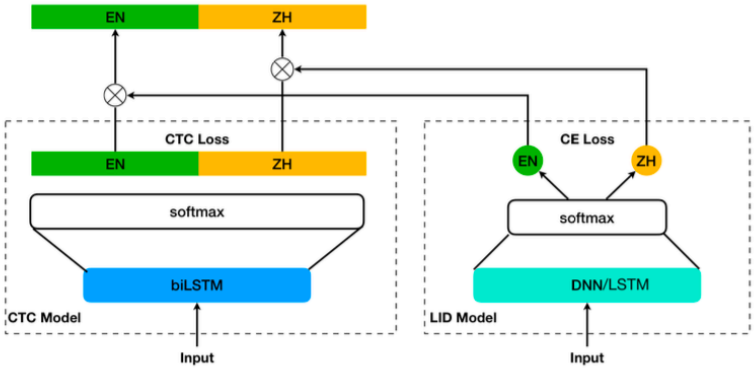

Figure 3.3.: CTC model with frame-level language classification.

architecture.
In their results, they achieve improvements over the baseline model without the additional language ID probabilities incorporated, even if there was no code-switching data seen during training. When training the models with code-switching data the authors show further improvements on their CS test set, this is however accompanied by a trade-off with the monolingual WER. In there experiments if a setting improves the code-switching WER it worsens the performance on the monolingual sets. This shows that there is the need for some more analysis in this approach as well, especially if it is desirable to be able to keep a high performance on monolingual data.

4. Approach

This chapter will describe the architectures used in this thesis. The data used for training and how it was used will be explained, too.

4.1. Models

Here a brief description of the models used will be given.

4.1.1. Base model

In this thesis, the basis of all models, is the seq2seq encoder-decoder based model, as described in [26]. An abstract display of the model is given in Figure 4.1. The model consists of an encoder and a decoder network. The encoder is made up of two two-dimensional CNN layers with 32 filters. The window size is three over both the time and the frequency dimensions. The stride with which the window is moved is defined as two for both dimensions. The outputs are than inputted into six bidirectional LSTM layers. As we do not have future outputs during inference of the model the decoder is made up of two LSTM layers which are not bidirectional. Afterwards the output of the decoder and encoder LSTMs are fed to the Multi-Head Attention network, which uses the decoder LSTM output as the query and the encoder output as the key and the value. This module than determines on which encoder frames to focus in order to predict the next output token. Afterwards the output goes through a linear layer followed by a residual connection depicted by the ”Add” bloc. This is then put into a linear layer, which projects the input to a dimension equaling the number of output tokens which are to be classified. Afterwards a logarithmic soft-max function is applied to get a probability distribution over the target labels. While the input of the encoder are the log-Mel spectrum's of the utterance, the input to the decoder are the embeddings of previously outputted tokens and the start of sentence token in case of the first decoding step. An abstract description of the neural network would look like:

$$
\begin{aligned}
enc &= \textit{Bi-LSTM}(\textit{CNN}(lMel\_Spectrum)) \\
tgt\_emb &= \textit{LSTM}(\textit{Embedding}(out\_tokens)) \\
dec &= (\textit{MHA}(enc, enc, tgt\_emb) + tgt\_emb) \\
output &= \textit{log\_softmax}(dec)
\end{aligned}
$$

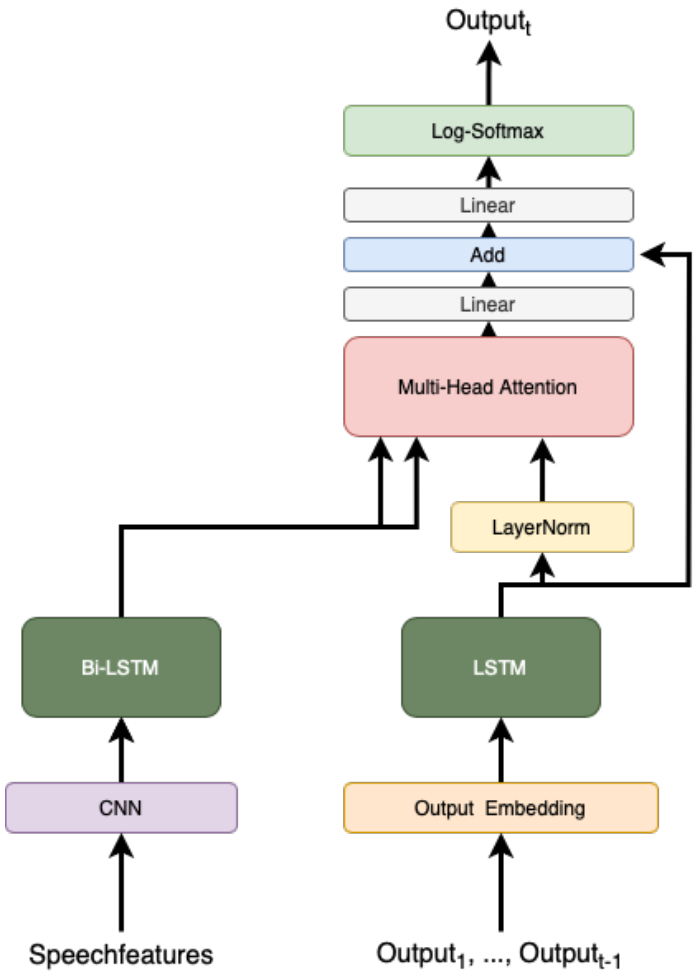

Figure 4.1.: Base sequence-to-sequence encoder-decoder model.

4.1.2. Different architectural additions

In this section, the different architectural additions to the baseline will be shown. In order to simplify the figures, the residual connections are not displayed.

- The first architecture we tried was a model which learns a token based language identifier next to predicting the next token of the target text. The architecture is shown in Figure 4.2 a).
- Figure 4.2 b) shows a multi-task learning approach similar to the first model. Instead of classifying the language IDs on the output of the decoder, a frame based language classification is applied on the output of the encoder.
- In order to further utilize the language prediction not only implicitly but also through incorporating this information, inspired by [14] the model shown in Figure 4.2 c) utilizes additional language dependent adapter modules. They modulate the output previous to the projection layer. Which adapter module to choose, is determined based on the current

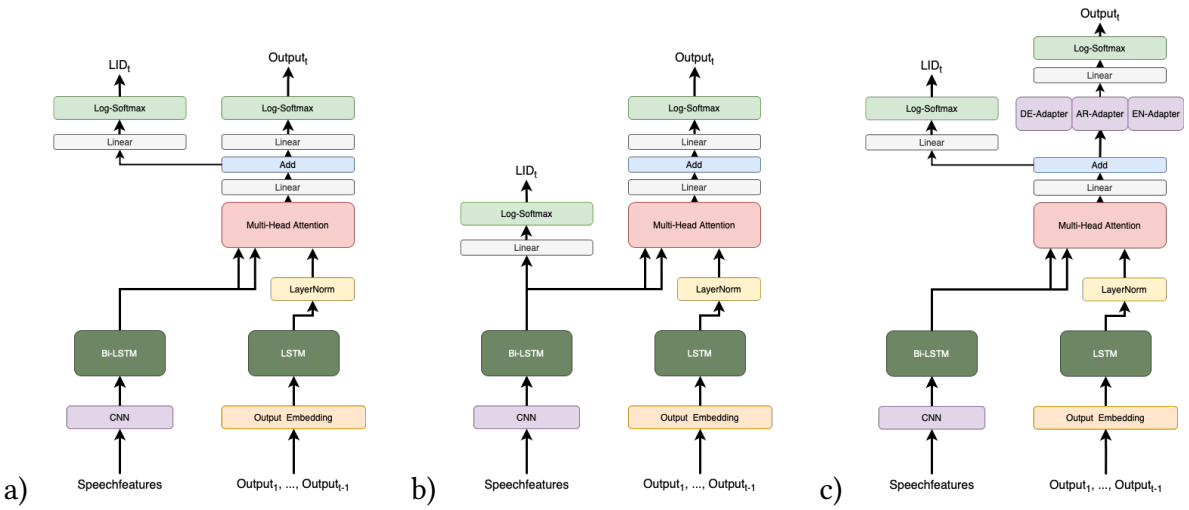

Figure 4.2.: a): Additional token-based language prediction. b): Additional frame-based language classification. c): Additional language classification which is used to determine the adapter module to be used.

language classification. The adapter modules consist of a down projection with a relu activation. This is followed by an up projection, which modulates the previous vector via a residual connection.

- Another way to utilize the already predicted language IDs is to add the language embedding on top of the output target embedding which can be regarded as a simplified version of [24] which was described in subsection 3.1.3. Figure 4.3 shows the architecture of suggested network.

4.2. Data

As this thesis is concerned with language-agnostic multilingual speech recognition, at first we started with generating a multilingual training set for the three languages used here, namely German, Arabic, and English.
The English set is made up of two public data sets.

- How2 [32]: This set contains speech in which explanations about different topics are given.
- TED-LIUM [31]: This is data collected from TED talks.

For the German data four corpora are used.

- Common Voice [3]: A data set consisting of general domain speech collected via Mozilla's Common Voice initiative.
- Europarl [16]: This corpus contains speeches collected from the European Parliament.

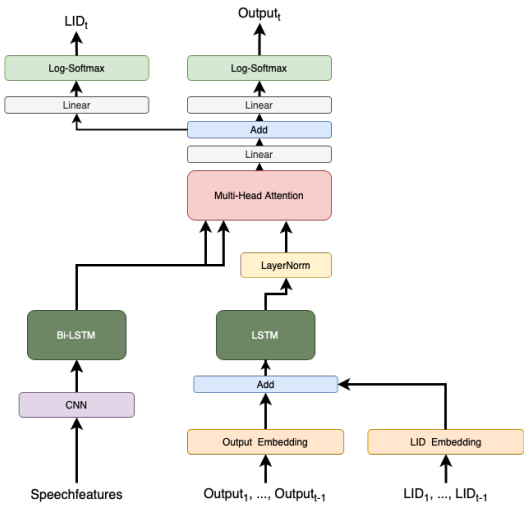

Figure 4.3.: Sequence-to-sequence encoder-decoder model, previous language prediction is used in following steps.

- Lectures: This is data collected from lectures given at the Karlsruher Institut für Technologie.
- Mini-international Neuropsychiatric Interview (MINI)-Data: A data set collected in cooperation with the ZI-Mannheim, containing recordings of MINI questions as well as answers.

The Arabic data consists of two corpora.

- Alj. 1200h [1]: This data set is made up of 1200 hours of broadcast videos taken from the Aljazeera Arabic TV channel. The speech is non-overlapping and mainly consists of modern standard Arabic (MSA) speech but utterances spoken with dialect are also present.
- MINI-Data: Data with Arabic speech of MINI questions and answers.

4.2.1. Data for training and evaluation

An overview of the data used during training is given in Table 4.1. In total, our data consists of 784h English, 865h German, and 1166h of Arabic transcribed speech. As will be described in section 4.3 we trained monolingual and multilingual systems. In order to keep the data settings used, as similar as possible, we decided to use all available data instead of reducing them to be similar in duration.
The English and German training data consist of many different speakers with a more free style of speaking compared to Arabic utterances. They also have more diverse recording settings especially considering the How2 set. The wast majority of the Arabic set, however, consists of speech which is read from a given text. As this data was collected from a news center it also has a very similar recording setting and most of the time a very high quality recording of the speech.

Language	Data-set	Utterance length	Number of utterances
English	How2	345h	210k
	TED-LIUM	439h	259k
German	Common Voice	314h	196k
	Europarl	46h	20k
	Lectures	504h	353k
	MINI-Data	1h	498
Arabic	Alj. 1200h	1127h	375k
	MINI-Data	39h	9k

Table 4.1.: Data used during training.

During training, the data sets are randomly combined in batches. Accordingly, a batch can contain utterances of all three languages.
As mentioned in subsection 2.4.2 a cross-validation data-set was used to calculate the improvements of the model and prevent effects like over-fitting. Table 4.2 shows the data distribution of the cross-validation sets. The cross-validation set is quite similar to the training data. One

Language	Data-set	Utterance length	Number of utterances
English	How2	10h	6k
	TED-LIUM	8h	5k
German	Common Voice	25h	15k
	MINI-Data	24min	173
Arabic	Alj. 1200h	9h	5k
	MINI-Data	2h	454
Inter-sentential	German, Arabic, English	53h	32k

Table 4.2.: Data used for cross-validation.

difference being that only Common Voice data was used in the case of German cross-validation. This is due to the nature of Common Voice covering a very general domain and as such yielding a good reflective data set. As the goal of this thesis is to enhance a models prowess on code-switching data, an artificially created inter-sentential data set is also used. This code-switching set is made up of all the other cross-validation data-sets. An example for such a code-switching sample is shown below.

”was heißt das auf englisch our customers want them”

This example utterance transcription is a concatenation of sentences from the German Common Voice and the English Wall Street Journal (WSJ) data sets. It is taken from the test data which is shown in Table 4.3.
As a means to evaluate the models, independent test data is needed. This data is not used for training or cross-validation. Table 4.3 depicts the data sets used for evaluating the models discussed in this thesis. Next to using a separate small set from TED-LIUM, about an hour of

Language	Data-set	Utterance length	Number of utterances
English	How2+TED-LIUM	3h	1k
	WSJ	1h	503
German	Common Voice	25h	15k
Arabic	Alj.	10h	5k
	Alj. 2h	2h	1k
Intra-sentential	Denglish	6min	89
Inter-sentential	Comm. Voice, Alj. 2h, WSJ	3h	2k

Table 4.3.: Test data to determine final performance of systems.

WSJ speech was used for evaluating the English performance. This set contains recordings of texts published in the Walt Street Journal. For the German language, a subset of the Common Voice was used. In the case of Arabic, two separate test sets are analysed. The Alj. set contains non-overlap MSA mixed with dialect data. The Alj. 2h set is a subset extracted from the Alj. 1200h corpus's test set which only contains MSA speech. Another set for evaluating intra-sentential code-switching was also used. The name Denglish describes the nature of this data which is German sentences with English words mixed in, for example.

"ich musste die harddisk neu formatieren weil der falsch gesteckte jumper
zur data corruption geführt hat und der computer gecrasht ist"

As it is visible in the illustration, Denglish utterances are mainly German text with only a few English words mixed in. Many times these words are additionally adapted to German grammar. The Inter-sentential data-set consists of code-switched utterances from the German Common Voice, Arabic Alj. 2h and the English WSJ data sets. The process of how exactly this set was generated is given in subsection 4.2.3.

4.2.2. BPE Data

The data-preparation applied to calculate the BPE tokens used in this thesis, is described in subsection 2.2.2. The German text used consisted of MINI text data, as well as Lecture, Common Voice, and Europarl text data. Additionally, a large corpus of only text was used. In subsection 2.2.2 we described taking only the first 100.000 lines of this shuffled text for our mixed text. In the case of determining the BPE tokens for the monolingual models, the first 300.000 lines are taken after having shuffled the language specific files. In the case of generating the text corpus for Arabic a big text-only corpus and MINI text data have been used. For English How2, TED-LIUM and another text-only corpus have been used. With an eye on having comparable results, a special emphasis was given on applying the same BPE generation steps for both the multilingual as well as the monolingual BPE generation for each language.

4.2.3. Inter-sentential code-switching

As we do not have a data set containing multilingual transcribed speech with code-switching we implemented an easy way of generating data artificially. Our general training data does

not contain an alignment between time-steps and exact words. The data only has aligned sentences with their corresponding time in the audio file. If we wanted to create intra-sentential code-switching data we would need to insert or append small utterances from one language to another utterance from another language, but due to the lack of data with word-time alignment this is a non-trivial task, as cutting out short sequences from a utterance could contain only parts of words or no speech at all, which in turn could lead to bad training data. This is why in this work only inter-sentential data was generated.
First, the total number of utterances from all sets are calculated, as they are all mixed into one data set in the case of training with no code-switching data. Afterwards depending on the expected percentage of code-switching data at least that amount of the original number of utterances is generated as code-switching utterances. In our experiments, we set this value to be 50%. As the original data consists of mainly utterances of 3-15 seconds length, $\frac{2}{8}^{\text{th}}$ of the 50% code-switching utterances are set to be less than five seconds, $\frac{2}{8}^{\text{th}}$ less than ten seconds, $\frac{2}{8}^{\text{th}}$ less than 15 seconds, $\frac{1}{8}^{\text{th}}$ less than 20 seconds and $\frac{1}{8}^{\text{th}}$ less than 25 seconds. Afterwards all the utterances, which have not been used for the code-switching data, are added to the training set. This is to avoid excluding utterances from training. If the training set has less utterances than the concatenation of the original monolingual sets, utterances randomly selected will be added to the training set as well. This is to make sure that only the specific percentage defined by the user are utterances with code-switching. The way utterances are selected for code-switching has following steps.

- Firstly the language of the utterance is determined. This is done randomly which means all languages have the same probability to be selected no matter the general size of the training data of the language itself. This can enable to have a more equal distribution of the number of samples between different languages during training.
- Afterwards a sample from the training set, of the previously selected language is drawn randomly. If this utterance exceeds the time limit it will be ignored otherwise it will be saved for the next step.
- The third step chooses a new language which must be different than the one of the last used sample.
- Next, a new sample from the language is drawn. If the total length of the previously saved utterances together with the newly selected one exceeds the time limit this sample gets discarded and the process is repeated beginning from step three. Otherwise, this sample is appended to the previously saved utterance.
- After adding two or more samples into one utterance the total length is checked against the time limit minus some offset which is set to two seconds. If it is longer than that value, the labels of the utterances are also concatenated accordingly and this newly created code-switching sample is added to the training set.
- Above steps are repeated for each time limit until each time the respective number of samples has been generated.

4.3. Experimental Setup

In this chapter, a description of our experimental setup is given. An explanation of how curriculum learning is utilized will be given as well. The implementation used in this thesis is written in Python [44] and leverages the PyTorch [28] library which uses the highly optimized CUDA [27] toolkit. This enables our models to utilize NVIDIA graphics processing units in an optimized way during training. By dropping frequency's and applying speed perturbations the model is trained to learn better generalizations of speech. Another effect is that the data augmentation artificially generates more data. These augmentations happen randomly in each epoch on different frequencies and the speed is also altered randomly. The features were calculated using a python and c++ [13] re-implementation of the feature calculation in the Janus Recognition Toolkit [47]. An example of such a feature can be seen in Figure 2.2. For training Adam optimization [15] is used. The adaptive learning rate depends on the number of warm-up steps which we defined as 8000 and the learning rate itself, here this is set to 0,002. For the first updates until the number of warm-ups is reached, the learning rate increases up to 0,002 during training. After that, the learning rate decreases linearly. The parameters of the five best models are saved after each epoch. The performance is determined on the cross-validation data set. If the model does not yield any improvements for five epochs, early abortion is applied. After each epoch the batches are also randomly shuffled, as otherwise there might be some form of over-fitting on long or short utterances depending on how the batches are sorted.

4.3.1. General Model Parameters

Unless otherwise mentioned the models are trained to classify 4000 target tokens. The input dimension is always 40-dimensional logarithmic Mel-Spectrum's, the specifics are explained in subsection 2.2.1. The number of neurons used in the encoder and decoder are 1024. All models utilize six bi-LSTMs in the encoder, and eight headed multi-head-attention mechanism with two LSTM layers in the decoder. During training a dropout of 20% is applied on the bi-LSTM outputs of the encoder except for the last one. The same dropout is applied on the output of the first LSTM in the decoder. Additionally a dropout of 20% was utilized on the embedding of the previously outputted targets which are used in the decoder.

4.3.2. Curriculum Learning

In this chapter, an explanation of how curriculum learning is utilized in training the models mentioned in section 4.1 will be given. During the first phase of training, only monolingual utterances are used. This means that even if the system is multilingual, each utterance in a batch contains speech of only one language. One batch, however, can consist of utterances from different languages. In our work, this step is referred to as the first-curriculum.
After the models are trained to saturation the so-called second-curriculum begins. In this step, the models are trained with data in which 50% of the utterances contain two or more languages. An explanation on how this is generated is given in subsection 4.2.3. In the second curriculum, an epoch of training is finished after seeing the total number of target tokens of all monolingual training sets. This is done in order to make training faster as the new training data with CS has overall longer sequences while keeping the same amount of samples.

5. Evaluation

In this chapter, the results of different experiments conducted for this thesis will be shown and discussed. All tests are conducted on the test data shown in Table 4.3 and are reported in WER%.

5.1. First-curriculum

This section will present the performance of different models trained on data which does not contain code-switching utterances.

5.1.1. Monolingual Baseline Models

In order to see how multilingual models compare to their monolingual counterparts a set of monolingual systems were trained and evaluated for German, Arabic, and English ASR. Table 5.1 shows the results of the monolingual models on monolingual test data.

model	How2+TED	WSJ	Alj.2h	Alj.	Comm.Voice	Average
Mono-De	98,15	91,04	131,36	128,42	**11,71**	92,14
Mono-Ar	108,08	100,46	**9,72**	**15,69**	102,47	**67,28**
Mono-En	**7,49**	**5,52**	177,29	170,92	143,82	101,01

Table 5.1.: Results of monolingual models on monolingual test sets.

As expected the results show that each model trained on one language has rather good performances on their respective test set. Interesting to note is that the Mono-De model which was trained solely on German training data was unable to predict anything on the How2+TED set. On the WSJ set which is more strict speech, the model sometimes was able to predict small parts of the utterance which is why the WER is lower in this case. As the example,

Reference: ”the papers comprise the public record of the pennzoil versus”
Hypothesis: ”der patriarch comprice the public recording of the pensional fr”

shows it mainly outputs German words or sub-words with similar sounds if any. Another interesting point is that the English words ”the public” were recognized correctly, which suggests that there has been some code-switching in the data which is supposed to only contain German speech. This might be due to the fact that using English words is becoming more and more common in the German language. On the Arabic test data the model outputs tokens with similar sounds which are completely wrong as the reference expects Arabic words written in

Arabic and as such the resulting WER is even higher than 100%. When evaluating German the WER is rather low. As mentioned in subsection 4.2.2 we tried to have the same setup for all models and used similar ways to create the BPE targets. One would expect the resulting monolingual labels to have a higher representational power concerning their language. In the case of the German BPE however, the text from which the tokens were generated did not contain any words with ”x” or ”q” which results in the model not being able to predict words containing this letters correctly. The ”x” are usually left out and ”q”’s are replaced by the letter ”u”.
Usually monolingual models output almost no hypothesis for utterances of different languages. The English model (Mono-En),however, outputs hypothesis for German and Arabic test sets which results in WER higher than 100%.

model	Denglish	Inter-sentential	Average
Mono-De	**16,67**	83,80	**50,24**
Mono-Ar	100,40	**73,33**	86,87
Mono-En	105,73	89,74	97,74

Table 5.2.: Results of monolingual models on code-switching test sets.

Table 5.2 shows results of the monolingual models on inter- & intra-sentential code-switching test sets. Interesting to see is that Mono-De has a rather good performance on Denglish data, which is reasonable, as these utterances are mainly German utterances with some English words mixed in. Depending on how common these English words are some can be predicted by this model. This also supports the idea that using English vocabulary while speaking is so common, that even our German training data contains such examples. An illustration of a correctly recognized Denglish sentence is:

”es gibt standing ovations von den zuschauern”

As can be seen in the table, the performance on the Inter-sentential tests is very poor for all models. This was to be expected after analysing the results of the monolingual models with respect to the test data from different languages. The Mono-De model has the overall best average WER over multilingual sets. However, this is mainly due to the performance on the Denglish data.

5.1.2. Multilingual Models

When training multilingual models we first trained a model with the same network and parameters as the monolingual models (Seq2seq). The only difference being the multilingual training data and the shared BPE labels. We also evaluated a baseline model in which the bi-LSTM outputs of the encoder are not added but used in a concatenated setting (Seq2seq_concat), important to note is that this setting also increases the number of learnable parameters. In Table 5.3 the model Seq2seq_lid describes the same model as the baseline (Seq2seq) with the only difference being that the output tokens are 4003 in which the three additional tokens

model	How2+TED	WSJ	Alj.2h	Alj.	Comm.Voice	Average
Seq2seq	9,95	8,47	11,10	17,61	18,31	13,09
Seq2seq_concat	8,98	6,81	10,68	16,31	16,33	11,82
Seq2seq_lid	8,85	7,69	10,92	17,06	16,99	12,30
Seq2seq_mt_dec	**8,18**	6,73	9,97	**15,93**	**14,86**	**11,13**
Seq2seq_mt_enc	8,91	6,96	10,49	16,66	16,54	11,91
Seq2seq_add_emb	8,49	**6,47**	**9,91**	15,94	15,47	11,26
Seq2seq_adapt	8,20	6,58	10,19	16,12	14,88	11,19

Table 5.3.: Results of multilingual models on monolingual test sets.

are language tags for each language. This model is trained to first predict the language tag at the beginning or when a switch happens in case of code-switching and then start predicting the transcription similar to the setting described in subsection 3.2.1. Seq2seq_mt_dec is the model depicted in Figure 4.2 a). Seq2seq_mt_enc is the model in Figure 4.2 b), Seq2seq_adapt is Figure 4.2 c) and Seq2seq_add_emb is the model shown in Figure 4.3. The best performances on each test set are shown in bold.

Table 5.3 shows that the baseline multilingual model trained with all training data randomly mixed, generally speaking, performs better taking the average over all monolingual test sets. However, considering the test data individually we see that the monolingual models outperform the multilingual ones on their respective test sets. The average relative WER increases on the English set by up to 41,58% and Arabic by 12,99% for the Seq2seq model. The performance of the model also decreases on the German data by 56,36% relative WER.

Another point is that the baseline model, concatenating the encoder outputs performs slightly better than the additive version, this is probably due to the additional number of learnable parameters. That being said, it still can not reach the performance of the monolingual models. As can be seen, all models trained with slightly different architectures outperform the additive baseline model and the models Seq2seq_mt_dec, Seq2seq_add_emb and Seq2seq_adapt even outperform the baseline using the concatenation of the bi-LSTM outputs, although they have fewer parameters. Another interesting point here is, that out of these three models the one with the fewest number of parameters has the best average WER, over the monolingual test sets, with 11,13%.

The most significant decrease in performance compared to the monolingual counterpart is seen for the German set. The Common Voice test data had the highest WERs of the monolingual models in Table 5.1 as well. Looking at Table 4.1 we see that the training data from the Common voice data set makes up only about 36% of all the German training data. Compared to the other languages, this mismatch between training and test data could be the reason for the overall worse WERs in the results. This effect seems to get enhanced, when additionally mixing in training data from the other languages. And thus the performance decreases especially for this test set.

The authors in [46] mentioned an increase in the CER performance when they used their Seq2seq_lid model compared to the monolingual models. However, next to having increased the number of parameters the authors have three differences when compared to our setting. The first is that they used a CTC-loss based system, secondly they use graphemes as labels

and last but not least they trained their multilingual model on at least seven languages. The Seq2seq_lid system in our setting with only three languages has only minor improvements of 6,04% relative WER over our baseline Seq2seq multilingual model but still performed worse than the monolingual systems. Comparing our results to the relative improvement of 10,57% CER and 16,74% in the case of more parameters mentioned in [46], this makes us believe that such a systems probably profits more if it is trained on more than just three languages. Another point to mention is that their improvements over the monolingual models are mainly due to the very poor CER performances on languages like Spanish which is reported with a CER of 50,8% for the monolingual model. This is reduced to 29,6% in the multilingual system.
Our model depicted in Figure 4.2 a) yields the best results and is able to reduce the average WER relative by 14,97%. In the case of the multilingual models it achieves best scores on How2+TED, Alj. and Common Voice sets. In contrast to WSJ and Alj.2h, these data mainly contain more challenging speech due to being free or dialect data. This suggests that the token based classification of the language restricts the model to focus on more detailed information of the context vector of the Multi-Head-Attention system, resulting in higher scores on these kind of utterances. The WERs on WSJ and Alj.2h are one of the best as well, which shows that this system has no focus shift towards the acoustic model as it still performs very good on speech following a more grammatically correct structure.
Another model improving over the baseline model is the Seq2seq_mt_enc system. Although having a quite similar structure to the Seq2seq_mt_dec system, the results show a decrease in performance on the Comm.Voice set. This suggests that only restricting the encoder to be able to identify the correct language is not enough of an extra restriction for the overall model. As the restrictions on the decoder show better improvements.
In our model which utilizes the previously predicted language ID in the decoder via additionally using its embedding (Seq2seq_add_emb), we see an improvement over both baseline models. Most interestingly though, are the results on the WSJ and the Alj.2h data sets. Here this system has the best WERs of all multilingual models, with 6,47% and 9,91% WER. This shows that the additional embedding of the language in the decoder is especially useful for transcribing utterances which follow a grammatically correct structure without many free speech elements. Although not having the best WER on any single test set, utilizing adapter modules, as in Seq2seq_adapt, clearly shows improvements and generally good results. This is supported by having the second best average WER 11,19%, over all monolingual test sets. This corresponds to a relative improvement of 14,51% WER over the baseline model.

model	Denglish	Inter-sentential	Average
Seq2seq	18,67	43,35	31,01
Seq2seq_concat	17,47	**37,71**	**27,59**
Seq2seq_lid	16,67	51,93	34,30
Seq2seq_mt_dec	**15,33**	49,38	32,36
Seq2seq_mt_enc	17,33	45,92	31,63
Seq2seq_add_emb	17,20	62,22	39,71
Seq2seq_adapt	16,93	49,44	33,19

Table 5.4.: Results of multilingual models on code-switching test sets.

Table 5.4 shows the results of our multilingual models on code-switching test sets. Similar to the results depicted in Table 5.3 the two baseline models achieve similar scores with the Seq2seq_concat version having a slight edge.
An interesting point, however, is the result on specifically the Denglish data. Both multilingual baselines have a decrease of the performance. The Seq2seq model decreases by 12,00% WER in relation to the monolingual German model. Although generally speaking the decreases for intra-sentential code-switching becomes more clear when looking at some examples.

Reference: ”lass uns ein paar ideen brainstormen”
Mono-De: ”lasst uns ein paar ideen brain stammen”
Seq2seq: ”lass uns ein paar ideen brainstormen”

This example shows that the multilingual model in contrast to the monolingual one is better able to predict English words like ”storm” in ”brainstormen”. Despite that the over all performance decreases due to examples like.

Reference: ”nach der prüfung chillen wir erst mal”
Mono-De: ”nach der prüfung chilen wir erst mal”
Seq2seq: ”nach der prüfung tschühen wir erstmal”

These examples suggest that the acoustic model of the system gets enhanced due to the more diverse phonemes in all languages. As can be seen in the second example this can lead to suboptimal results. Specifically the second example could be due to some sort of bias of the decoder for tokens which were seen during training for the currently decoded language. This hypothesis will be further evaluated in section 5.2.
The best performance on the Denglish data is achieved by the Seq2seq_mt_dec model with a WER of 15,33%. This corresponds to a relative improvement of 17,89% WER over the multilingual baseline model and 8,04% WER over the monolingual German system. This model yields further improvements in correctly predicting English tokens.

Reference: ”wir müssen einen teil des konzerns downsizen”
Mono-De: ”wir müssen einen teil des konzerns downseisen”
Seq2seq: ”wir müssen einen teil des konzerns down zeisen”
Seq2seq_mt_dec: ”wir müssen einen teil des konzerns downsizen”

In the case of this example, however, the model did not predict the language ID for English and kept on classifying German as the language. This might be due to the argument maximum which is applied to get the language ID with the highest probability at each decoding step. As such this does not show if the probability for English was just short of the German ID. Moreover looking at the overall language ID prediction we see that the model only predicts the German language ID for the whole Denglish set. This is probably due to fact that during training the model never saw that an utterance could contain multiple languages and as such has a bias to predict the language ID based on the token embeddings previously outputted, due to the nature of the LSTM used in the decoder. This holds true for all models which apply a multi-task learning.

On the Inter-sentential data, all models struggle to predict especially the long sequences. This is probably because such long utterances are heavily underrepresented in the training data. While it seems that systems like Seq2seq_adapt and Seq2seq_mt_dec have an easier time predicting languages IDs than Seq2seq_mt_enc or Seq2seq_lid as can be seen in the following example transcription results as well.

Reference: "في النهاية أيها الأعزآء يبقى السؤل hat frankfurt mehr einwohner als gießen"

Seq2seq: "في نهاية أيها والعزآء يبقى السؤل هتفانك فوتنيا عين هنا أسأل"

Seq2seq_mt_dec: "في نهاية أيها الأعزآء يبقى السؤل hat frankfurt mehr einwohner als gießen"

Seq2seq_adapt: "في نهاية أيها الأعزآء يبقى السؤل hat frankfurt mehr einwohner als gießen"

Seq2seq_mt_enc: "في نهاية أيها الأعزآء يبقى السؤل"

Seq2seq_lid: "في نهاية أيها الأعزآء يبقى السؤل"

There are still many cases in which all systems struggle to predict code-switching when decoding the utterance. Looking at the following hypothesis below. A common appearance is, that only one language is predicted and the utterance of the other one is just left out in the transcription.

Reference: "من مجموع زيت النخيل في العالم these powers are important to us"

Seq2seq: "من مجموع زيت الناحية في العالم"

Seq2seq_mt_dec: "من مجموع زيت النحيل في العالم"

Seq2seq_mt_enc: "من مجموع زيت النحيل في العالم"

Seq2seq_lid: "من مجموع زيت النحيلي في العالم"

This happens especially often in the case of code-switching utterances between Arabic and German or English. In some cases with code-switching, the systems generate hypotheses for speeches of both languages. However, as can be seen in the following example they decode tokens for one language only.

Reference: "wie alt bist du they are the products of a success profile"

Seq2seq: "the ordges do they are the products of a success profile"

Seq2seq_mt_dec: "the artist war they are the products of a success profile"

Looking at the predicted language ID the model Seq2seq_dec_mt only predicted the English language and accordingly tried to decode an English transcription. This is observed with all models predicting languages. They struggle to predict more than one language per utterance. The results of all models after training the first-curriculum can be seen in Table A.1 in the appendix.

5.2. Second-curriculum

In this chapter, the models presented in subsection 5.1.2 are trained a second time as described in section 4.3. After the systems were trained to saturation the parameters which yielded the

lowest perplexity on the cross-validation sets are taken and trained a second time, but this time with utterances containing code-switching.
As we are more concerned with different architectural set-ups and the effects of using manually generated code-switching data, we only used the Seq2seq model as our baseline for the second-curriculum. Comparing the best models performance from the first-curriculum with

model	How2+TED	WSJ	Alj.2h	Alj.	Comm.Voice	Average
Seq2seq	7,13	5,46	9,43	**14,93**	13,28	10,05
Seq2seq_lid	7,23	**5,33**	9,39	15,12	**13,05**	10,02
Seq2seq_mt_dec	**6,97**	5,44	**9,25**	15,02	13,11	**9,96**
Seq2seq_mt_enc	7,25	5,77	9,39	15,26	13,19	10,17
Seq2seq_add_emb	7,40	5,54	9,40	15,30	13,10	10,15
Seq2seq_adapt	7,47	5,70	9,70	15,18	13,32	10,27

Table 5.5.: Results of multilingual models on monolingual test sets after second-curriculum.

11,13% WER in Table 5.3 to the new WER of 9,96% in Table 5.5, in which 50% of the utterances seen during training contain code-switching, a relative improvement of 10,51% WER can be observed.
A very important observation, however, is that using only code-switching without any architectural changes yields very impressive improvements even on the baseline model. Whereas the original baseline had a WER of 13,09% the same model after training the second-curriculum has 10,05% WER which is a relative improvement of 23,22%. Another point is that the overall discrepancy between the performance of the baseline model and the different architectures is closer after the second-curriculum, which indicates that this training regime is especially useful and helps for a better generalisation in the case of the baseline architecture. Having a better generalisation in the acoustic model results in better predictions in cases with difficult pronunciations which otherwise might get confused with more commonly appearing words, such as the word ”unscathed” as shown in the example below.

Reference: ”is about emerging on the other side of a challenging experience unscathed or unmarked by the experience”
First-Curriculum Seq2seq: ”is about emerging on the other side of a challenging experience unscaped or unmarked by the experience”
Second-Curriculum Seq2seq: ”is about emerging on the other side of a challenging experience unscathed or unmarked by the experience”

Another interesting result is that the sentence ”thank you thanks” was only predicted correctly by the Seq2seq and Seq2seq_adapt models.

Reference: ”thank you thanks”
Seq2seq: ”thank you thanks”
Seq2seq_lid: ”right here because”
Seq2seq_mt_dec: ”here thanks”
Seq2seq_mt_enc: ”here thanks”
Seq2seq_add_emb: ”here thanks”
Seq2seq_adapt: ”thank you thanks”

This suggests that the additional multi-task loss can lead to worse results although the language was classified correctly. The adapter modules, however, are able to learn some additional information in order to re-correct the probability distribution. An interesting example which might yield some light on why the Seq2seq_mt_dec model has the best results on the How2+TED set is the following sample.

Reference: ”sixty thousand”
Seq2seq: ”sechzig tausend”
Seq2seq_lid: ”sixty thousand”
Seq2seq_mt_dec: ”sixty thousand”
Seq2seq_mt_enc: ”sechzig phase”
Seq2seq_add_emb: ”sechzig tausend”
Seq2seq_adapt: ”sixty thousand”

This example shows interesting results and the effects of different architectures. Firstly we see that most models confused the English utterance with a German one and as such predicted German tokens. Interestingly all models which predict language IDs start predicting a German ID except for Seq2seq_lid which correctly predicts English this suggests that this model has an approach looking into a broader context when predicting the language than the other multi-task systems. Secondly only the Seq2seq_mt_dec model is able to predict the English language after first wrongfully predicting German for the first few tokens. Accordingly these two models are able to decode the utterance correctly. To support the hypothesis attention heads with the broadest attention on the first token prediction are shown in Figure 5.1. In the right image it is possible to see that the the model which first predicts the language itself has a higher attention around the 9th and also the 22th frames. In contrast image a) mainly focuses on the first four frames. It can be seen that the Seq2seq_lid focuses on more frames when predicting the first token which is always a language identifier. The Seq2seq_mt_dec model instead focuses on fewer frames in order to predicting the first target token. Another point is that although predicting the wrong language the adapter modules are still able to decode the correct sequence which could be due to some CS data between English and German, which seems to be present in the German training corpus as was deduced in subsection 5.1.1.
Comparing the best model’s average WER of 6,21% on the English sets with the 6,51% WER of the Mono-En model from Table 5.1 we see a relative improvement of 4,61%.

Looking at the average WER of 12,14% over the Arabic test sets compared to 12,71% of the Mono-Ar a relative improvement of 4,48% WER can be seen.
In the case of testing on German test data, we see a deterioration of the performance from 11,71% WER for the Mono-De model to 13,11% WER. As mentioned in, subsection 5.1.2, this might be to the generally more difficult test data and the general data distribution in the German training data.

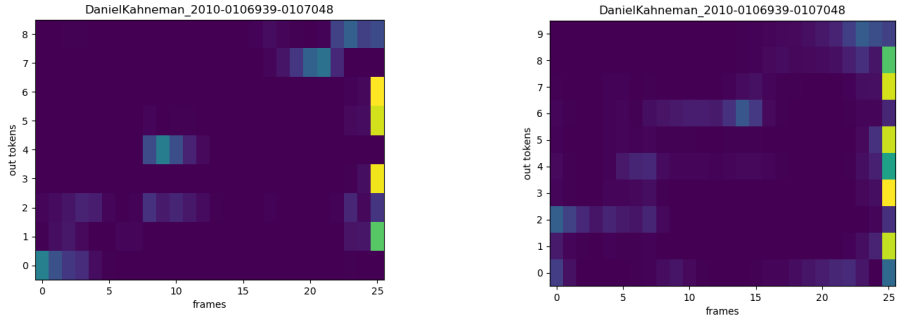

Figure 5.1.: Attention of the most significant head for predicting the first token of the utterance ”sixty thousand” with the identifier: DanielKahneman_2010-0106939-0107048. a) Left figure: Seq2seq_mt_dec. b) Right figure: Seq2seq_lid.

Table 5.6 shows results of multilingual models trained with 50% code-switching on CS test data. Maybe the most striking result is the Inter-sentential WER result of the Seq2seq_add_emb.

model	Denglish	Inter-sentential	Average
Seq2seq	14,80	8,98	11,89
Seq2seq_lid	**14,13**	9,02	**11,58**
Seq2seq_mt_dec	14,53	**8,77**	11,65
Seq2seq_mt_enc	14,53	9,05	11,79
Seq2seq_add_emb	15,07	59,92	37,50
Seq2seq_adapt	15,20	9,06	12,13

Table 5.6.: Results of multilingual models on code-switching test sets after second-curriculum.

With 59,92% WER, it is by far the highest error. Analysing the following example utterance, we can see that while other models were able to transition from predicting Arabic towards

German this model failed to do so.

Reference: ”ما حدث في جنوب السودان يعني شيء بشع sie schaden uns”

Seq2seq: ”ما حدث في جنوب السودان يعني شيء بشع sie schaden uns”

Seq2seq_lid: ”ما حدث في جنوب السودان يعني شيء بشع sie schaden uns”

Seq2seq_mt_dec: ”ما حدث في جنوب السودان يعني شيء بشع sie schaden uns”

Seq2seq_mt_enc: ”ما حدث في جنوب السودان يعني شيء بشع sie schade uns”

Seq2seq_add_emb: ”ما حدث في جنوب السودان يعني شيء بشع sie sie sie

نتخابات ما حدث في جنوب السودان يعني شيء بشع sie sie sie

... وجه ما حدث في جنوب السودان يعني شيء بشعsieعلن”

Seq2seq_adapt: ”ما حدث في جنوب السودان يعني شيء بشعsie schaden uns”

In the above example, the prediction for the Seq2seq_add_emb model was cut off as it kept going on in a similar fashion. Analysing the transcription hypothesis as well as the predicted language IDs of this model it shows that the model is not able to predict the second language after starting to predict Arabic. This might be caused, due to the reason that the language embedding of the previously predicted language is added on the previously outputted token embedding. If the model is not able to predict the correct language when the switch from one language to the other happens it is not able to break the bias towards that one language which is being decoded. Another reason for this model performing poorly on this data set is that during the decoding step a beam search is only applied on the outputted BPE tokens and not on the language IDs as well, and as such only the language with the highest probability at each step is used in the following steps, which can worsen the results.
Although the other results are all quite close to each other, with a WER of 8,77% the Seq2seq_mt_dec model outperforms all other models and improves the baseline by about 2,34% relative WER. Comparing it to the best result from the first-curriculum systems which was the Seq2seq_concat model with a WER of 37,71% we can see a relative improvement of 76,74% WER.
In the case of the Denglish test set, the best performing model is Seq2seq_lid. It shows a relative improvement of 4,53% WER over the baseline model with 14,80% WER. In the following example utterance, all models predicting a language classified German as the LID. However, it is possible to see that models which aim at being more aware of code-switching get confused when decoding ”wann hast du”.

Reference: ”sag mal wann hast du dir diesen schönen oldtimer gekauft”

Seq2seq: ”sag mal wann hast du dir diesen schönen oldtimer gekauft”

Seq2seq_lid: ”sag mal wannners du dir diesen schönen oldtimer gekauft”

Seq2seq_mt_dec: ”sag mal wannna studiert diesen schönen oldtimer gekauft”

Seq2seq_mt_enc: ”sag mal wann er studiert diesen schönen oldtimer gekauft”

Seq2seq_add_emb: ”sag mal wann hast du dir diesen schönen oldtimer gekauft”

Seq2seq_adapt: ”sag mal wannna so dir diesen schön oldtimer gekauft”

It seems that systems which have an additional task on focusing on the language spoken, have a less strict language model or focus too much on the acoustic model, which can lead to

mistranscriptions. The Seq2seq_add_emb however is able to predict the correct sequence. In this case, having the language as an additional embedding seems to be an advantage over the other models. Listening and looking at other utterances it is possible to infer that this model is able to enhance the language model and is able to produce good results although the utterance was spoken unclear and are partially mumbled. However, in cases where an English word is expected to be decoded during the mainly German speech this effects the prediction negatively. With a WER of 14,13% the Seq2seq_lid model performs best. The following example suggests why this might be the case.

Reference: ”wir müssen einen teil des konzerns downsizen”
Seq2seq: ”wir müssen einen teil des konzerns downs heißen”
Seq2seq_lid: ”wir müssen einen teil des konzerns downsizen”
Seq2seq_mt_dec: ”wir müssen einen teil des konzerns down sizen”
Seq2seq_mt_enc: ”wir müssen einen teil des konzerns down seisen”
Seq2seq_add_emb: ”wir müssen einen teil des konzerns downs eisen”
Seq2seq_adapt: ”wir müssen einen teil des konzerns downes eisen”

Interestingly the only model which predicted a switch of languages in the utterance was the Seq2seq_mt_dec one. Technically this system transcribed the speech correctly however, it did not integrate the English word into the German grammar. In contrast, the Seq2seq_lid was able to do that. This suggests that in the case of intra-sentential code-switching this model might be preferable as it only slightly restricts the transcription towards a language but still is able to decode words of different languages as it has to watch out for potential code-switching cases. Not having predicted an actual code-switching might be due to our training data, in which the frames present after switching are larger in numbers and as such the model might not predict the actual switch.

model	Total Average
Seq2seq	10,57
Seq2seq_lid	10,47
Seq2seq_mt_dec	**10,44**
Seq2seq_mt_enc	10,63
Seq2seq_add_emb	17,96
Seq2seq_adapt	10,80

Table 5.7.: Average WER of multilingual models on all test sets after second-curriculum.

The average WERs over all test sets is given in Table 5.7. We can see that the overall best performing model is the Seq2seq_mt_dec closely followed by the Seq2seq_lid.
Table A.2 in the appendix shows the results of all models after the second-curriculum.

5.3. Additional Experiments

A few additional experiments were also conducted to further analyse our experiments. The Seq2seq_lid_emb model is a multilingual model which gets the language embedding as additional information during the decoding. As such this model can not be considered language-agnostic. In order to analyse the effect of different code-switching ratios, the Seq2seq baseline with only 20% code-switching utterances was tried as well (Seq2seq_0.2). To further look into the influence of code-switching data during training the baseline model was also trained with the same ratio of 50% but the data did not contain any transition from Arabic to German utterances (Seq2seq_noarde). These two models where also trained with curriculum learning. To investigate the effect of applying curriculum learning a baseline model Seq2seq_nocurr was trained with 50% of the utterances containing code-switching without applying a two step training. Table 5.8 shows the WERs of the above mentioned experiments to see the effect of

model	How2+TED	WSJ	Alj.2h	Alj.	Comm.Voice	Average
Seq2seq	7,13	5,46	9,43	14,93	13,28	10,05
Seq2seq_lid_emb	7,87	**5,09**	9,86	15,87	13,42	10.42
Seq2seq2_nocurr	8,26	7,16	10,74	16,57	15,42	11,63
Seq2seq_0.2	**6,91**	5,24	**9,34**	**14,86**	**12,75**	**9,82**
Seq2seq_noarde	7,07	5,59	9,64	15,28	13,26	10,17

Table 5.8.: Results of multi-lingual models on mono-lingual tests using different training set ups.

different code-switching settings as well as our reference model, which is not language-agnostic, on monolingual test data.
An important result we can conclude from these evaluations is that applying a curriculum during training yields significant improvements over not using it. This is supported by the WERs of the Seq2seq_nocurr model which are higher than all other models. With an average WER of 11,63% the performance, compared to the Seq2seq result with 10,05% WER a relative decrease of 18,38% WER can be seen.
Another very interesting conclusion which can be drawn is that the system which gets the language embedding as additional information in the decoder is outperformed on all test tasks by the language-agnostic models which were trained in a curriculum fashion. The only test set in which the Seq2seq_lid_emb model performed better is the WSJ data. This suggests that this model and the additional language embedding mainly benefit for very formal and grammatically correct utterances as this is mainly the case in the WSJ set, whereas the other corpora are primarily made up of free- or dialectal speech.
Even though the Seq2seq_noarde model had no transitions from Arabic speech to German it is interesting to see that the model does not suffer the same problems the models had after only training the first-curriculum. In section 5.1 it was shown that the models struggled to predict the transcription of another language after starting to decode in one language. This model however, although having never seen a transition from Arabic to German is able to decode

utterances with such a code-switching case. This is shown in the example below.

Reference: ”متدنية الإنسان حقوق حالة عندك umso größer ist ihre beliebtheit in der türkei”
Seq2seq_noarde: ”متدنية الإنسان حقوق حالة عندك ما umser größer ist ihre belebtheit in der türkei”

The most striking result however is achieved by the Seq2seq_0.2 model. Training the baseline model has shown a significant improvement over the other language-agnostic models. It is only outperformed by the Seq2seq_lid_emb model on the WSJ set which is probably due to its very formal nature. The average WER decreases from 10,05% WER to 9,82% which is a relative enhancement of 2,31% WER over the Seq2seq model trained with 50% code-switching data. This result shows that the exact portion of code-switching data needs to be further elaborated as it is a hyperparameter. The improvement could be the result of how the code-switching data is generated. As it is trained to generate shorter utterances containing multiple languages some utterances get reused multiple times. This in return reduces the diversity of the utterances seen during training. Reducing the portion of code-switching data however, might have a better balance between sequences with multiple languages while keeping the data more diverse. Another aspect is that even using less than 50% code-switching data is able to break the bias towards one language which was seen in the evaluation of the first-curriculum results section 5.1.

model	Denglish	Inter-sentential	Average
Seq2seq	14,80	8,98	11,89
Seq2seq_lid_emb	14,53	77,22	45,88
Seq2seq2_nocurr	16,27	10,50	13,39
Seq2seq_0.2	**13,73**	**8,62**	**11,18**
Seq2seq_noarde	15,07	9,14	12,11

Table 5.9.: Results of multi-lingual models on code-switching tests using different training set ups.

In table 5.9 the results of the same models on code-switching test sets are depicted. As the Seq2seq_lid_emb model needs to have the correct language embedding at each decoding step which is not possible during inference the language ID used for the Denglish set was the German one as these utterances are mainly German speech, section 4.2. For the Inter-sentential data the language ID of the first utterance was taken.
The Seq2seq_lid_emb has a very good performance of 14,53% WER on the Denglish data which is probably due to the reason that as we previously inferred, in subsection 5.1.1 the monolingual German data already contains some English words. The Denglish utterances being mainly German words and following German grammar rules thus might be handled by this system more easily. However, as one might expect the WER on the Inter-sentential data is extremely bad when compared to the other models.
Again similar to the results on the monolingual sets, training without curriculum learning significantly worsens the models WER on all test data. The average 13,39% WER shows a

relative degradation of 12,62%WER compared to the Seq2seq model trained with 50% code-switching data in the second-curriculum.
Although Seq2seq_noarde was given 50% code-switching utterances only missing out on transitions from Arabic to German a slight degradation on both Denglish and Inter-sentential data can be seen. This could probably be due to the reason that by eliminating this transition the general amount of German and Arabic data gets slightly reduced and as such the Seq2seq system is moderately better on both test sets especially on the Denglish one in which German speech is an essential part. The training set of the Seq2seq model had German speech in 47% percent of all samples whereas the Seq2seq_noarde only had it in 42,85% of the samples.
As already seen in Table 5.8 again the baseline model trained with only 20% code-switching data outperforms all other models. Here the average WER is improved by 8,44% WER relatively over the baseline model which again shows that the portion of how much training data is made up of multiple languages is an important hyperparameter.
To account for an easier comparison with other models a table with all results is given in Table A.3 in the appendix.

6. Conclusion and Future Work

6.1. Conclusion

In this work, we looked into the task of code-switching by utilizing language-agnostic multilingual speech recognition systems. We evaluated the intra- & inter-sentential code-switching performance in the context of the three languages Arabic, German and English. We also tested our models on monolingual sets to consider a potential performance degradation on such data as well. For the intra-sentential tests, we used a small in-house data set which contained Denglish data which is mainly German sentences with English words mixed in. In case of the inter-sentential data we generated our own artificial corpora based on data sets of the three languages as described in subsection 4.2.3. We used the same algorithm to generate a training set containing code-switching data as well.
Several experiments for developing multilingual models were conducted. We trained different model architectures with different tasks. Models were trained with and without utilizing the artificial code-switching data. In order to compare our models, we strictly applied the same processes in all experiments. Monolingual models were also trained and their performances were evaluated and compared to the multilingual ones. Additionally a non language-agnostic multilingual model was trained and evaluated on our test sets.

- In our experiments we show that plainly training a language-agnostic multilingual model decreases performance on monolingual utterances when compared to their respective monolingual counterpart models. The performance on CS data is also underwhelming.
- We show that utilizing code-switching data during training significantly improves the WER over the baseline and should be applied when training multilingual models.
- The models heavily benefit from applying a curriculum learning regime when training with code-switching data, as it improves their performance remarkably.

We demonstrate that the above mentioned methods not only significantly improve the performance on code-switching data but also improve on the English and Arabic monolingual tests as well. Additional to our baseline system we also explored multiple different architectural additions as well as predictional changes.

- One model has additional three language IDs in its target set and predicts one of them whenever a new language is decoded.
- In another model we adopted a multi-task learning approach in which the feature vector used to predict the target token is also used to predict the language ID for each token.

- The third approach used the output of the encoder to predict the language in a frame-based manner.
- We also implemented two systems which explicitly utilized language predictions. One of them applied a language specific adapter module on the decoder output based on the language which was classified in the current step.
- The second model utilized the token based language prediction by autoregressively adding its embedding on the previously outputted token embedding.

Our results show improvements of these models on the intra-sentential code-switching test when trained without data containing code-switching. However, after training the second-curriculum as described in section 5.2 all models improve and there is only a relative improvement of 1,23% WER over the baseline model when using the above mentioned token based multi-tasking model, which is depicted in Figure 4.2 a).
Strikingly, after training the second-curriculum, all our language-agnostic models outperform the multilingual reference model which is given the correct language embedding at all decoding steps. This again underlines the importance of showing the model code-switching data during training even if it was generated artificially. In section 4.3 we also analysed using a smaller portion of code-switching data. This further improved our results which suggests that this a hyperparameter where the optimal value needs still to be determined.

6.2. Future Work

Currently, we are collecting inter-sentential CS test data fur the purpose of having a more expressive evaluation set to work with.
In the future, we would like to explore the effect of training a massively multilingual monolithic system with our code-switching data and see the effects on such a system. We hope that this training regime could even be beneficial for languages with fewer data in such setups and could alleviate problems caused by underrepresented languages when training multilingual systems. Interesting to observer would be if the models still improve over the monolingual baselines even in a heavily multilingual setting as well.
As the results for the baseline model with only 20% code-switching data have shown improvements over the model with 50% we further need to analyse the best portion as well as invest in researching a more enhanced algorithm for creating this kind of data. One improvement of the algorithm could be to limit the number of times a specific utterance is allowed to be used for creating the data containing multiple languages.
In [20] the authors showed that including code-switching data increases the frame based language ID classification accuracy by a significant amount. Generating intra-sentential code-switching utterances where the task of transcription is left out and only training to classify the language might help in the case of the transcribing speech which contains intra-sentential code-switching. This could potentially lead to worse results, in the case of Denglish as well, as the grammar of the main language usually needs to be applied to the transcription even if it is a foreign word.
In their work in [23], the writers showed that using additional features such as the pitch and the

fundamental frequency variation feature yielded impressive improvements when transcribing tonal languages, such as Vietnamese or Chinese. In our system the encoder implicitly learns to extract information important for the transcription task. Adding a tonal language into our multilingual setup could further enhance the prowess of our acoustic model, as none of the three languages used in this thesis are tonal.
Another approach could be to utilize a text auto-encoder which is trained with either synthetic or natural code-switching text-only data. In this scenario the decoder would be shared with the ASR system and as such would decode intra- & inter-sentential code-switching during training which could lead to a more flexible and sophisticated decoder.

Bibliography

[1] Ahmed Ali, Peter Bell, James Glass, Yacine Messaoui, Hamdy Mubarak, Steve Renals, and Yifan Zhang. "The MGB-2 challenge: Arabic multi-dialect broadcast media recognition". In: *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2016, pp. 279–284.

[2] Djegdjiga Amazouz, Martine Adda-Decker, and Lori Lamel. "Addressing code-switching in French/Algerian Arabic speech". In: *Interspeech 2017*. 2017, pp. 62–66.

[3] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. "Common voice: A massively-multilingual speech corpus". In: *arXiv preprint arXiv:1912.06670* (2019).

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. "Curriculum learning". In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 41–48.

[6] Joyce YC Chan, PC Ching, and Tan Lee. "Development of a Cantonese-English code-mixing speech corpus". In: *Ninth European Conference on Speech Communication and Technology*. 2005.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[8] Jeffrey L Elman. "Finding structure in time". In: *Cognitive science* 14.2 (1990), pp. 179–211.

[9] Philip Gage. "A new algorithm for data compression". In: *C Users Journal* 12.2 (1994), pp. 23–38.

[10] Helen Gremmelmaier. "Multilingual Sequence-To-Sequence Speech Recognition". PhD thesis. Karlsruhe Institute of Technology, 2021.

[11] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[12] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[13] ISO. *ISO/IEC 14882:1998: Programming languages — C++*. Available in electronic form for online purchase at http://webstore.ansi.org/ and http://www.cssinfo.com/. Sept. 1998, p. 732. URL: http://webstore.ansi.org/ansidocstore/product.asp?sku=ISO%2FIEC+14882%2D1998;%20http://webstore.ansi.org/ansidocstore/product.asp?sku=ISO%2FIEC+14882%3A1998;%20http://www.iso.ch/cate/d25845.html;%20https://webstore.ansi.org/.

[14] Anjuli Kannan, Arindrima Datta, Tara N Sainath, Eugene Weinstein, Bhuvana Ramabhadran, Yonghui Wu, Ankur Bapna, Zhifeng Chen, and Seungji Lee. "Large-scale multilingual speech recognition with a streaming end-to-end model". In: *arXiv preprint arXiv:1909.05330* (2019).

[15] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[16] Philipp Koehn et al. "Europarl: A parallel corpus for statistical machine translation". In: *MT summit*. Vol. 5. Citeseer. 2005, pp. 79–86.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[19] *LeNet-5Arch*. https://www.topbots.com/important-cnn-architectures/. Accessed: 2021-08-30.

[20] Ke Li, Jinyu Li, Guoli Ye, Rui Zhao, and Yifan Gong. "Towards code-switching ASR for end-to-end CTC models". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 6076–6080.

[21] Dau-Cheng Lyu, Tien-Ping Tan, Eng-Siong Chng, and Haizhou Li. "An analysis of a Mandarin-English code-switching speech corpus: SEAME". In: *Age* 21 (2010), pp. 25–8.

[22] Rajend Mesthrie. *Introducing sociolinguistics*. Edinburgh University Press, 2009.

[23] Florian Metze, Zaid AW Sheikh, Alex Waibel, Jonas Gehring, Kevin Kilgour, Quoc Bao Nguyen, et al. "Models of tone for tonal and non-tonal languages". In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE. 2013, pp. 261–266.

[24] Markus Müller, Sebastian Stüker, and Alex Waibel. "Language adaptive multilingual CTC speech recognition". In: *International Conference on Speech and Computer*. Springer. 2017, pp. 473–482.

[25] Saida Mussakhojayeva, Yerbolat Khassanov, and Huseyin Atakan Varol. "A Study of Multilingual End-to-End Speech Recognition for Kazakh, Russian, and English". In: *International Conference on Speech and Computer*. Springer. 2021, pp. 448–459.

[26] Thai-Son Nguyen, Sebastian Stueker, Jan Niehues, and Alex Waibel. "Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 7689–7693.

[27] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. *CUDA, release: 10.2.89*. 2020. URL: https://developer.nvidia.com/cuda-toolkit.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[29] Ngoc-Quan Pham, Jan Niehues, Thanh-Le Ha, and Alex Waibel. "Improving zero-shot translation with language-independent constraints". In: *arXiv preprint arXiv:1906.08584* (2019).

[30] Shana Poplack. "Sometimes i'll start a sentence in spanish y termino en espanol: toward a typology of code-switching1". In: (1980).

[31] Anthony Rousseau, Paul Deléglise, and Yannick Esteve. "TED-LIUM: an Automatic Speech Recognition dedicated corpus." In: *LREC*. 2012, pp. 125–129.

[32] Ramon Sanabria, Ozan Caglayan, Shruti Palaskar, Desmond Elliott, Loïc Barrault, Lucia Specia, and Florian Metze. "How2: a large-scale dataset for multimodal language understanding". In: *arXiv preprint arXiv:1811.00347* (2018).

[33] Ernst Günter Schukat-Talamazzini. "Automatische Spracherkennung : Grundlagen, statistische Modelle und effiziente Algorithmen". PhD thesis. Braunschweig [u.a.], 1995. ISBN: 3528054921.

[34] Tanja Schultz and Alex Waibel. "Experiments on cross-language acoustic modeling." In: *INTERSPEECH*. 2001, pp. 2721–2724.

[35] Tanja Schultz and Alex Waibel. "Language-independent and language-adaptive acoustic modeling for speech recognition". In: *Speech Communication* 35.1-2 (2001), pp. 31–51.

[36] Hiroshi Seki, Shinji Watanabe, Takaaki Hori, Jonathan Le Roux, and John R Hershey. "An end-to-end language-tracking speech recognizer for mixed-language speech". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 4919–4923.

[37] Sanket Shah, Basil Abraham, Sunayana Sitaram, Vikas Joshi, et al. "Learning to recognize code-switched speech without forgetting monolingual speech recognition". In: *arXiv preprint arXiv:2006.00782* (2020).

[38] Changhao Shan, Chao Weng, Guangsen Wang, Dan Su, Min Luo, Dong Yu, and Lei Xie. "Investigating End-to-end Speech Recognition for Mandarin-english Code-switching". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 6056–6060. DOI: 10.1109/ICASSP.2019.8682850.

[39] Changhao Shan, Chao Weng, Guangsen Wang, Dan Su, Min Luo, Dong Yu, and Lei Xie. "Investigating end-to-end speech recognition for mandarin-english code-switching". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 6056–6060.

[40] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[41] Richard Skiba. "Code switching as a countenance of language interference". In: *The internet TESL journal* 3.10 (1997), pp. 1–6.

[42] Sebastian Stuker, Tanja Schultz, Florian Metze, and Alex Waibel. "Multilingual articulatory features". In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03)*. Vol. 1. IEEE. 2003, pp. I–I.

[43] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks". In: *CoRR* abs/1409.3215 (2014). arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

[44] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[45] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang. "Phoneme recognition using time-delay neural networks". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.3 (1989), pp. 328–339. DOI: 10.1109/29.21701.

[46] Shinji Watanabe, Takaaki Hori, and John R Hershey. "Language independent end-to-end architecture for joint language identification and speech recognition". In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE. 2017, pp. 265–271.

[47] Monika Woszczyna, Naomi Aoki-Waibel, Finn Dag Buo, Noah Coccaro, Keiko Horiguchi, Thomas Kemp, Alon Lavie, Arthur McNair, Thomas Polzin, Ivica Rogina, et al. "JANUS 93: Towards spontaneous speech translation". In: *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing*. Vol. 1. IEEE. 1994, pp. I–345.

[48] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. "Comparative study of CNN and RNN for natural language processing". In: *arXiv preprint arXiv:1702.01923* (2017).

[49] Shuai Zhang, Jiangyan Yi, Zhengkun Tian, Jianhua Tao, and Ye Bai. "Rnn-transducer with language bias for end-to-end Mandarin-English code-switching speech recognition". In: *2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE. 2021, pp. 1–5.

A. Appendix

A.1. Evaluation tables

model	How2+TED	WSJ	Alj.2h	Alj.	Comm.Voice	Denglish	Inter-sentential	Average
Mono-De	98,15	91,04	131,36	128,42	**11,71**	**16,67**	83,80	80,16
Mono-Ar	108,08	100,46	**9,72**	**15,69**	102,47	100,40	73,33	**72,88**
Mono-En	**7,49**	**5,52**	177,29	170,92	143,82	105,73	89,74	100,07
Seq2seq	9,95	8,47	11,10	17,61	18,31	18,67	43,35	18,21
Seq2seq_concat	8,98	6,81	10,68	16,31	16,33	17,47	**37,71**	**16,33**
Seq2seq_lid_emb	**7,87**	**5,09**	**9,86**	**15,87**	**13,42**	**14,53**	77,22	20,55
Seq2seq_lid	8,85	7,69	10,92	17,06	16,99	16,67	51,93	18,59
Seq2seq_mt_dec	8,18	6,73	9,97	15,93	14,86	15,33	49,38	17,20
Seq2seq_mt_enc	8,91	6,96	10,49	16,66	16,54	17,33	45,92	17,54
Seq2seq_add_emb	8,49	6,47	9,91	15,94	15,47	17,20	62,28	19,39
Seq2seq_adapt	8,20	6,58	10,19	16,12	14,88	16,93	49,44	17,48

Table A.1.: Results of all models after training without code-switching data. Best WERs of mono-& multilingual models are written in bold.

model	How2+TED	WSJ	Alj.2h	Alj.	Comm.Voice	Denglish	Inter-sentential	Average
Seq2seq_noCurr	8,26	7,16	10,74	16,57	15,42	16,27	10,50	12,13
Seq2seq	7,13	5,46	9,43	14,93	13,28	14,80	8,98	10,57
Seq2seq_0.2	**6,91**	**5,24**	9,34	**14,86**	**12,75**	**13,73**	**8,62**	**10,21**
Seq2seq_lid	7,23	5,33	9,39	15,12	13,05	14,13	9,02	10,47
Seq2seq_mt_dec	6,97	5,44	**9,25**	15,02	13,11	14,53	8,77	10,44
Seq2seq_mt_enc	7,25	5,77	9,39	15,26	13,19	14,53	9,05	10,63
Seq2seq_add_emb	7,40	5,54	9,40	15,30	13,10	15,07	59,92	17,96
Seq2seq_adapt	7,47	5,70	9,70	15,18	13,32	15,20	9,06	10,80
Seq2seq_noarde	7,07	5,59	9,64	15,28	13,26	15,07	9,14	10,72

Table A.2.: Results of all models after training with code-switching data. Best WER performances are written in bold.

model	How2+TED	WSJ	Alj.2h	Alj.	Comm.Voice	Denglish	Inter-sentential	Average
Mono-De	98,15	91,04	131,36	128,42	**11,71**	**16,67**	83,80	80,16
Mono-Ar	108,08	100,46	**9,72**	**15,69**	102,47	100,40	73,33	**72,88**
Mono-En	**7,49**	**5,52**	177,29	170,92	143,82	105,73	89,74	100,07
Seq2seq	9,95	8,47	11,10	17,61	18,31	18,67	43,35	18,21
Seq2seq_concat	8,98	6,81	10,68	16,31	16,33	17,47	**37,71**	**16,33**
Seq2seq_lid_emb	**7,87**	**5,09**	**9,86**	**15,87**	**13,42**	**14,53**	77,22	20,55
Seq2seq_lid	8,85	7,69	10,92	17,06	16,99	16,67	51,93	18,59
Seq2seq_mt_dec	8,18	6,73	9,97	15,93	14,86	15,33	49,38	17,20
Seq2seq_mt_enc	8,91	6,96	10,49	16,66	16,54	17,33	45,92	17,54
Seq2seq_add_emb	8,49	6,47	9,91	15,94	15,47	17,20	62,28	19,39
Seq2seq_adapt	8,20	6,58	10,19	16,12	14,88	16,93	49,44	17,48
Seq2seq_noCurr	8,26	7,16	10,74	16,57	15,42	16,27	10,50	12,13
Seq2seq	7,13	5,46	9,43	14,93	13,28	14,80	8,98	10,57
Seq2seq_0.2	**6,91**	**5,24**	9,34	**14,86**	**12,75**	**13,73**	**8,62**	**10,21**
Seq2seq_lid	7,23	5,33	9,39	15,12	13,05	14,13	9,02	10,47
Seq2seq_mt_dec	6,97	5,44	**9,25**	15,02	13,11	14,53	8,77	10,44
Seq2seq_mt_enc	7,25	5,77	9,39	15,26	13,19	14,53	9,05	10,63
Seq2seq_add_emb	7,40	5,54	9,40	15,30	13,10	15,07	59,92	17,96
Seq2seq_adapt	7,47	5,70	9,70	15,18	13,32	15,20	9,06	10,80
Seq2seq_noarde	7,07	5,59	9,64	15,28	13,26	15,07	9,14	10,72

Table A.3.: Results of all models upper part without lower part with inter-sentential CS data seen during training. Best WERs of models trained with and without CS are written in bold. Values of best performing monolingual models are written in bold as well.