FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Peter Polák

Spoken Language Translation via Phoneme Representation of the Source Language

Institute of Formal and Applied Linguistics

Supervisor of the master thesis:	doc. RNDr. Ondřej Bojar, Ph.D.
Study programme:	Computer Science
Study branch:	Artificial Intelligence

Prague 2020

Dedication.

Title: Spoken Language Translation via Phoneme Representation of the Source Language

Author: Peter Polák

Institute: Institute of Formal and Applied Linguistics

Supervisor: doc. RNDr. Ondřej Bojar, Ph.D., Institute of Formal and Applied Linguistics

Abstract: We refactor the traditional two-step approach of automatic speech recognition for spoken language translation. Instead of conventional graphemes, we use phonemes as an intermediate speech representation. Starting with the acoustic model, we revise the cross-lingual transfer and propose a coarse-to-fine method providing further speed-up and performance boost. Further, we review the translation model. We experiment with source and target encoding, boosting the robustness by utilizing the fine-tuning and transfer across ASR and SLT. We empirically document that this conventional setup with an alternative representation not only performs well on standard test sets but also provides robust transcripts and translations on challenging (e.g., non-native) test sets. Notably, our ASR system outperforms commercial ASR systems.

Keywords: spoken language translation automatic speech recognition transfer learning non-native speech translation

Contents

Introduction **4**

1 Tasks, Architectures, and Datasets **5**
1.1 Automatic Speech Recognition 5
1.1.1 Brief History 5
1.1.2 Models 5
1.2 Spoken Language Translation 7
1.3 Neural Network Architectures and Models 8
1.3.1 Jasper 8
1.3.2 QuartzNet 8
1.3.3 Transformer 11
1.4 Data Representation 13
1.4.1 MFCC 13
1.4.2 Text Representation in NMT 15
1.5 Data Sets 16
1.5.1 LibriSpeech 16
1.5.2 Common Voice 16
1.5.3 Large Corpus of Czech Parliament Plenary Hearings 16
1.5.4 CzEng 1.7 17
1.6 Error Metrics 18
1.6.1 Word Error Rate 18
1.6.2 BLEU 18

2 Automatic Speech Recognition **19**
2.1 Coarse-to-Fine Intermediate Step and Cross-Lingual ASR Transfer . . 19
2.1.1 Introduction 19
2.1.2 Related Work 20
2.1.3 Data and Models Used 21
2.1.4 Examined Configurations 21
2.1.5 Results and Discussion 25
2.1.6 Conclusion and Future Work 25
2.1.7 Note on the Results 26
2.2 Speech Recognition to Phonemes 27
2.2.1 Related Work 27
2.2.2 Experiment Set-Up 28
2.2.3 Training 28
2.2.4 Evaluation 30
2.2.5 Conclusion and Future Work 32

3 Enhanced ASR **34**
3.1 Related Work 34
3.1.1 ASR 34
3.1.2 Text Encoding 36
3.2 Experiment Motivation and Outline 37
3.3 Word Encoding Selection 37

3.3.1 Experiment Outline . . . 38
3.3.2 Data Preparation . . . 38
3.3.3 Training . . . 40
3.3.4 Results and Analysis . . . 40
3.3.5 Conclusion . . . 43
3.4 English and Czech Enhanced ASR . . . 43
3.4.1 Experiment Outline . . . 43
3.4.2 Training Configuration . . . 44
3.4.3 Performance Evaluation . . . 45
3.4.4 Conclusion . . . 47

4 Spoken Language Translation 48
4.1 Related Work . . . 48
4.2 Motivation and Experiment Outline . . . 48
4.3 Training . . . 49
4.4 Evaluation . . . 49
4.4.1 SLT Test Set Preparation . . . 50
4.4.2 Beam Size . . . 51
4.4.3 Czech to English . . . 51
4.4.4 English to Czech . . . 56
4.5 Conclusion . . . 59

5 Fine-tuning Enhanced ASR 60
5.1 Related Work . . . 60
5.2 English "Corrupted" Training Data . . . 60
5.2.1 Data Preparation . . . 61
5.2.2 Ensemble Training . . . 61
5.2.3 Acoustic Models Performance . . . 62
5.2.4 English Corrupted Data Collection . . . 62
5.3 Czech "Corrupted" Training Data . . . 63
5.3.1 Task Setup . . . 63
5.3.2 Ensemble Training . . . 64
5.3.3 Acoustic Models Performance . . . 64
5.3.4 Czech Corrupted Data Collection . . . 65
5.4 Fine-Tuning of Enhanced ASR . . . 65
5.4.1 Experiment Outline . . . 65
5.4.2 Training/Fine-Tuning . . . 67
5.4.3 Performance Evaluation . . . 67
5.5 Conclusion and Future Work . . . 69

6 Online Adaptation 70
6.1 Enhanced ASR/SLT Pipeline with Online Adaptation . . . 70
6.1.1 Online Adaptation Model . . . 70
6.2 Experiments . . . 72
6.2.1 Naïve Approach: Train & Rewrite All . . . 72
6.2.2 Revised Naïve Approach: Train & Rewrite All Word-by-Word 74
6.2.3 Embedded Word Beam Search . . . 75
6.2.4 Analysis . . . 76
6.3 Conclusion . . . 77

Conclusion **79**

Bibliography **81**

List of Figures **88**

List of Tables **89**

List of Abbreviations **90**

A Attachments **91**
A.1 Electronic Attachment Contents 91

Introduction

Spoken language translation (SLT) is an area of natural language processing that gained much popularity in recent years. The technology has many uses in practical scenarios, such as business meetings, conferences, or in multi-national organizations (e.g., EU). Currently, human interpreters are employed, but deep learning brings new possibilities in attempts to automate the task.

The contemporary SLT system consists of standalone automatic speech recognition (ASR) followed by a machine translation (MT). More recent studies review end-to-end SLT models. The advantage of end-to-end solutions is their technical simplicity and better preservation of nonverbal information contained in the speech. On the other hand, their obvious limitation is the requirement of "end-to-end" training datasets, i.e., source language voice recordings paired with the translated transcripts in the target language. Such datasets are only a little available and not available at all for most language pairs.

The promising idea is to refactor the traditional two-step approach of ASR of the source language and MT of the text into the target language. The idea is to shorten the ASR step, finishing only with a string of phonemes, and translate directly from source-side phonemes to the target text.

In this thesis, we explore SLT with the outlined change: speech recognition into phonemes and translation from phonemes in the source to target words. For this, we build an SLT framework with an intermediate phoneme-level transcription step for the translation between Czech and English in both directions.

We start from the acoustic model. The challenge for this part is to overcome the scarcity of Czech training data. We review a transfer from English and propose a coarse-to-fine technique. This technique can be used combined with the transfer learning method, but we demonstrate that alone performs even better. Subsequently, we deal with the proposed change of the output representation in phonemes. To speed-up the training, we use a transfer from conventional ASR with the "adaptation phase" proposed in the coarse-to-fine method.

Further, we continue with the translation model. We deal with practical considerations such as input and output encoding. We experiment with different model sizes and also enhancing the model performance with the ability to correct some of the errors produced by the acoustic model.

Finally, we experiment with speaker adaptation on the fly.

Thesis organization

We settled on a less traditional strategy for the organization of this work: in every chapter, we review the related work rather than in a standalone chapter. We review mutual related work in Chapter 1 (such as neural network architectures or corpora).

Chapters progressively deal with problematic starting with automatic speech recognition in Chapter 2, further enhancing ASR in Chapter 3, continuing with spoken language translation in Chapter 4. In Chapter 5, we review techniques making the ASR pipeline more robust. Adaptation to speaker is described in Chapter 6. Finally, we conclude the work in Section 6.3.

1. Tasks, Architectures, and Datasets

In this chapter, we give a brief overview of the theoretical foundations for this thesis. First, we define the main tasks that are part of the solution. Further, we describe the utilized neural network architectures. Lastly, we describe used data sets and introduce evaluation metrics.

1.1 Automatic Speech Recognition

Automatic Speech Recognition (ASR) is one of the most popular tasks in NLP. With ever-growing technology that becomes more and more integrated with our day-to-day life, it is clear that ASR will play an essential role in this process.

In this section, we first briefly review the history and then describe the most popular contemporary approaches to ASR.

1.1.1 Brief History

The first attempts for ASR systems stem from the 1950s. Early ASR systems worked with syllables, vowels, and phonemes [Juang and Rabiner, 2005]. For example, a spoken digit recognizer from Bell laboratories. Their system estimated formant frequencies (as a vowel is pronounced, vocal tract sounds with natural modes of resonance called a formant) of vowel regions of an isolated digit.

A big leap in ASR systems was the introduction and popularization of Hidden Markov models (HMMs) in the 1980s. HMMs caused architecture to shift from pattern recognition to statistical modeling. The success of HMM-based ASR systems continues even today in the form of hybrid models consisting of HMMs and either Gaussian Mixture Models (GMMs) or Artificial Neural Networks (ANNs).

In the last few years, deep neural networks (DNNs) gained much popularity. In many tasks, ranging from image processing to natural language processing, DNNs outperformed other known methods. Besides their performance, they tend to require less expertise and engineering skills for a particular task than other methods. This makes them available for researchers in many areas. DNNs are becoming a standard in ASR currently, but the first attempts were already made briefly after the introduction of the backpropagation algorithm [Rumelhart et al., 1986]. They were used for recognition of phonemes [Waibel et al., 1989] or short sequences of words [Lubensky, 1988]. Further important milestones for ASR were recurrent neural networks and attention (applied, for example, in Chorowski et al. [2014]).

1.1.2 Models

In this section, we introduce contemporary models utilized for ASR. There are two such approaches: (1) HMM-based models, (2) End-to-End neural models.

HMM-based Models

HMMs (Hidden Markov Models) are probably the most used technique in ASR [Padmanabhan and Johnson Premkumar, 2015, Yu and Deng, 2016]. Hidden Markov Models capture the speech sequence as sequence of oservations. In the ASR context, HMMs utilize either GMMs or ANNs in order to model frame-based speech features (features with no temporal information). Practitioners often refer to the HMM-based models combined with neural networks as hybrid models.

A typical pipeline of an HMM-based model works as follows: the input is first pre-processed and converted to features, typically the MFCCs (see Section 1.4.1). These features are further passed to the estimator. Common unit in a HMM-based pipeline is *phone*. The decoder employs an *acoustic model*, a *dictionary* and a *language model* to decode the speech. The acoustic model estimates the probability of the acoustic sequence given word sequence. ANNs or GMMs represent the acoustic model. Dictionary maps phone sequences to words. Finally, the language model estimates the apriori probability of word sequence, independent of the observed sound. Commonly, n-gram language models are used.

End-to-End Models

The alternative to the HMM-based models are End-to-End (E2E) ASR pipelines. The common trait for these models is their ability to produce the final, human-readable text given acoustic features using one end-to-end deep neural network. Recent advancements (such as Amodei et al. [2016], Li et al. [2019a]) clearly show the future direction towards this kind of ASR. The advantage of E2E models is that they require less engineering than HMM models. An open problem remains the higher requirement of training data.

The input of the model are commonly MFCC features extracted from speech data. E2E models use two approaches: Connectionist temporal classification (CTC) [Graves et al., 2006] loss (e.g., Hannun et al. [2014], Zhang et al. [2017]) or attention mechanism (e.g., Bahdanau et al. [2016]). The inference algorithms include a conventional beam search with the language model (LM) (again, an n-gram LM is used) that re-scores the beams during the decoding. The use of LM in the beam search further enhances the transcription quality.

1.2 Spoken Language Translation

Spoken Language Translation is another NLP task dealing with speech processing. In literature, the SLT task describes translation into a target language text. Commonly, SLT can also be a speech-to-speech translation. Traditional layout of speech translation, before the emergence of end-to-end systems, consisted of a speech recognition unit followed by machine translation. With the upraise of deep learning, E2E frameworks that do not need intermediate transcription step are gaining popularity [Bérard et al., 2016, 2018, Jia et al., 2019].

Direct comparison of both approaches to date seems inconclusive [Sperber et al., 2019]. Also, the E2E and cascaded SLT differ: E2E SLT must be trained on corpora consisting od source speech and target text ("E2E corpora"), while the cascading can be trained on independent corpora of speech and transcripts for ASR and independently parallel corpora for MT. This makes the latter suitable in cases where there is no corpus for a given language pair available (which is mostly the case). On the other hand, cascading speech recognition and machine translation for SLT often introduces errors in source language transcription that are further propagated to final translation.

1.3 Neural Network Architectures and Models

In this section, we introduce neural network architectures and models we engage in our work. The first two models, Jasper and QuartzNet, are used as acoustic models. Finally, we present Transformer, which serves as a translation and transcript correction model.

1.3.1 Jasper

Jasper [Li et al., 2019a] is a family of end-to-end, deep convolutional neural network ASR architectures.

Architecture Overview

The input of the model are MFCCs (Mel Frequency Cepstrum Coefficients, see Section 1.4.1) obtained from 20 ms frames with 10 ms stride. We use 64 features. The model outputs the probability over a given vocabulary for every processed frame. In our ASR pipeline, the vocabulary is IPA phonemes.

The input is passed through one pre-processing layer, followed by the central part of the network. Finally, tree post-processing layers are applied. The central part of the model consists of so-called "blocks".

The Jasper model consists of B blocks and R sub-blocks. Jasper authors introduce a naming convention where such model is described as "Jasper BxR". In our experiments, we use Jasper 10x5.

Each sub-block applies operations as follows: 1D convolution, ReLU activation, and dropout. All sub-blocks of a block have the same number of output channels.

Further, Jasper's authors observed that models deeper than Jasper 5x3 require residual connections to converge. Residual connections inspired by DenseNet [Huang et al., 2017] and DenseRNet [Tang et al., 2018] are employed.

The input of a block is connected to its last sub-block via residual connections. Because the number of channels differs, 1x1 convolution is applied to account for this. After this projection, batch normalization is applied. The output is then added to the output of the batch normalization layer in the last sub-block. Afterward, the activation function and dropout are used, producing the output of the current block.

The schema of Jasper with residual connections is provided in Figure 1.1. The biggest used configuration for English graphemes has 333 million parameters.

1.3.2 QuartzNet

QuartzNet [Kriman et al., 2019] is another end-to-end ASR architecture used in our work. QuartzNet is a convolutional neural network based on Jasper [Li et al., 2019a] architecture having a fraction of parameters (18.9 million versus 333 million) while still achieving near state-of-the-art accuracy.

Same as for the Jasper model, the network's input is 64 MFCC features computed from windows of length 20 ms and overlap 10 ms. The network outputs probability over the given alphabet for each time frame. For training is used CTC loss and for decoding beam search.

The main difference between QuartzNet and Jasper is the application of 1D time-channel separable convolutions. Such convolutions can be separated into 1D depthwise convolutional layers with kernel K and a pointwise convolution operating across all

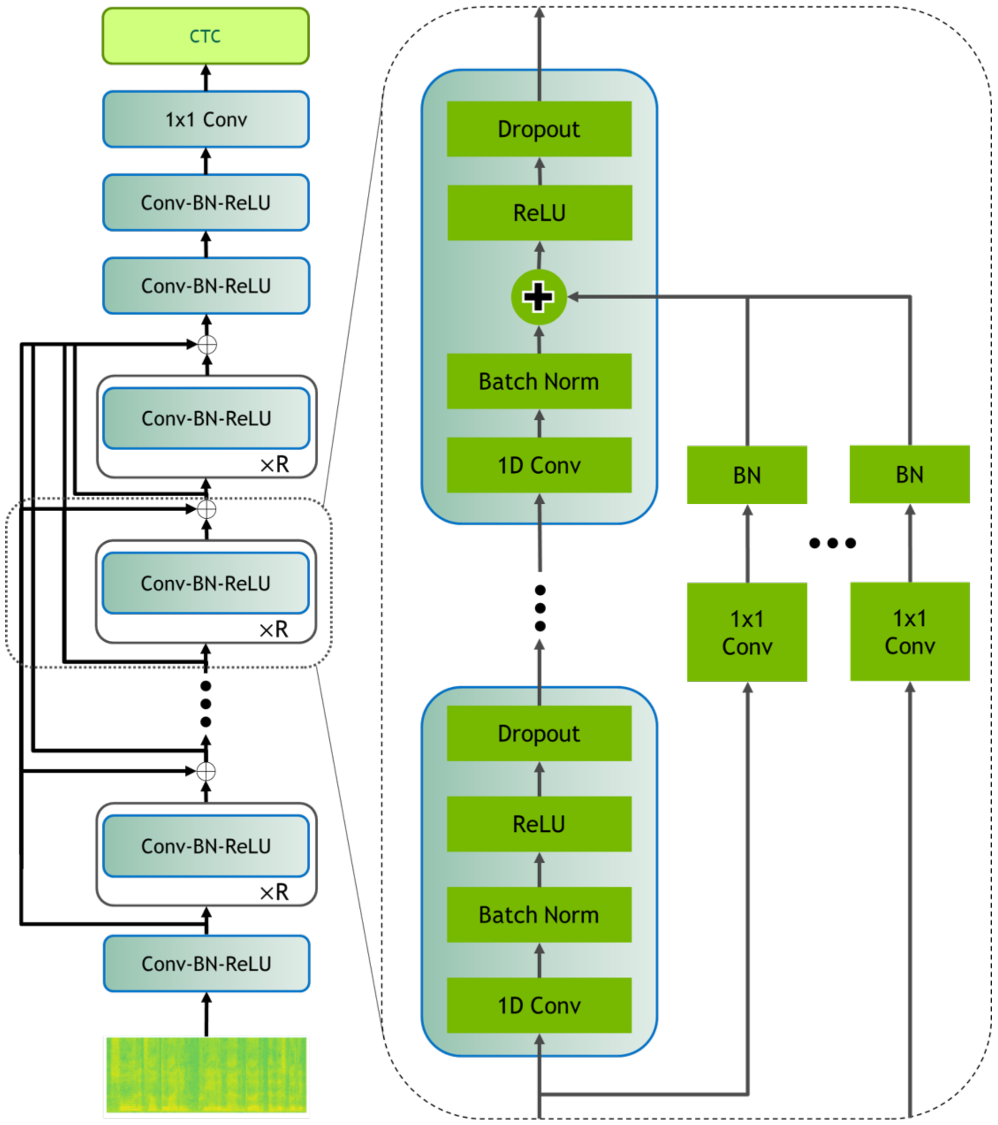


Figure 1.1: Jasper Dense Residual, taken from Li et al. [2019b].

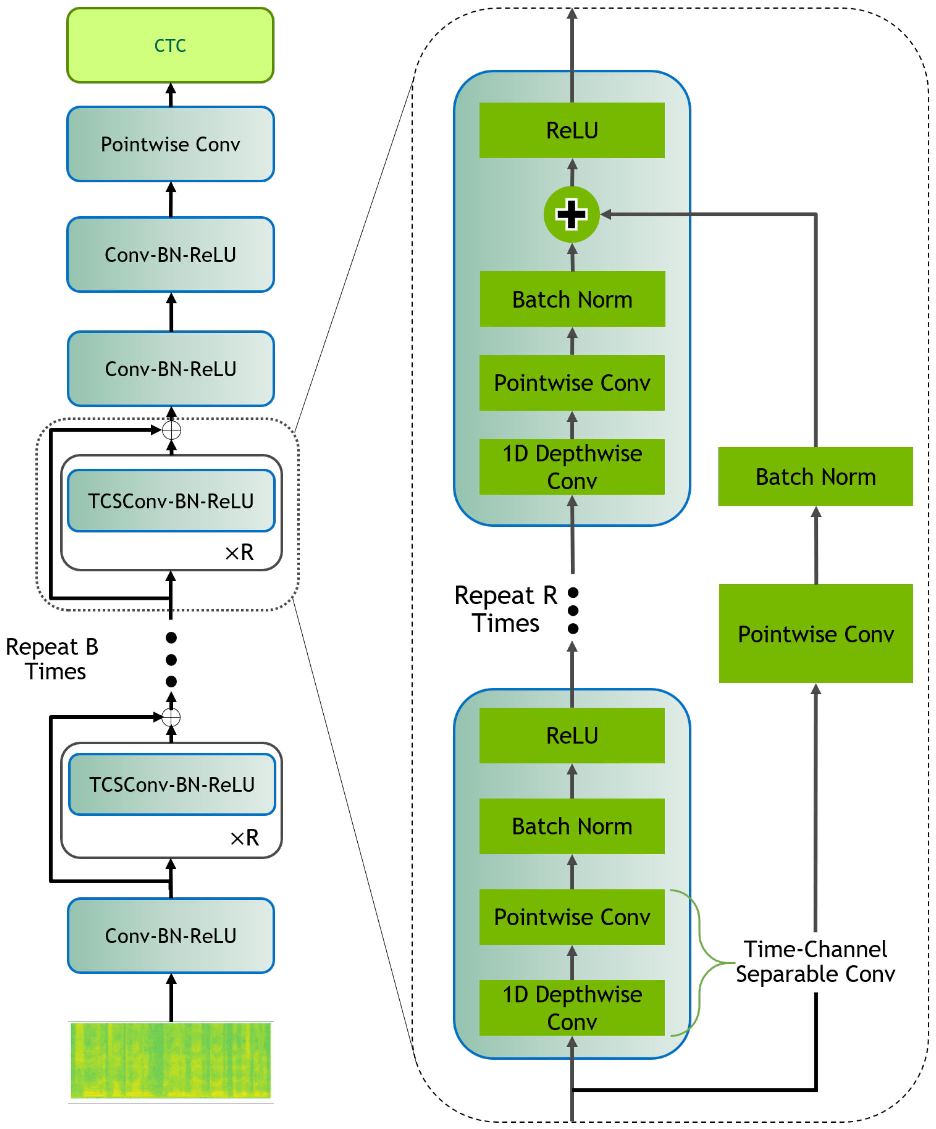


Figure 1.2: QuartzNet BxR architecture. Taken from Kriman et al. [2019].

channels on each time frame independently. A standard 1D convolutional layer with kernel size K and c_{in} input and c_{out} output channels has $K \times c_{in} \times c_{out}$ weights. On the other hand, the 1D depthwise convolution with kernel K has $K \times c_{in}$ weights followed by the pointwise convolution with $c_{in} \times c_{out}$ weights — together yielding $K \times c_{in} + c_{in} \times c_{out}$ weights. Because of this, the model has fewer parameters and can even have $3\times$ larger kernel than the bigger Jasper model. The architecture schema is in Figure 1.2.

The authors describe a further reduction of weights by using grouped pointwise convolution instead of the pointwise convolution layer (see Figure 1.3). When using four groups, the number of parameters is halved (for QuartzNet-15x5 from 18.9M to 8.7M) with slightly worse performance (WER 3.98 increases to 4.29 for LibriSpeech dev clean and 11.58 increases to 13.48).

As the first described weights reduction is sufficient for our purposes, we work with QuertzNet without the grouped pointwise convolution.

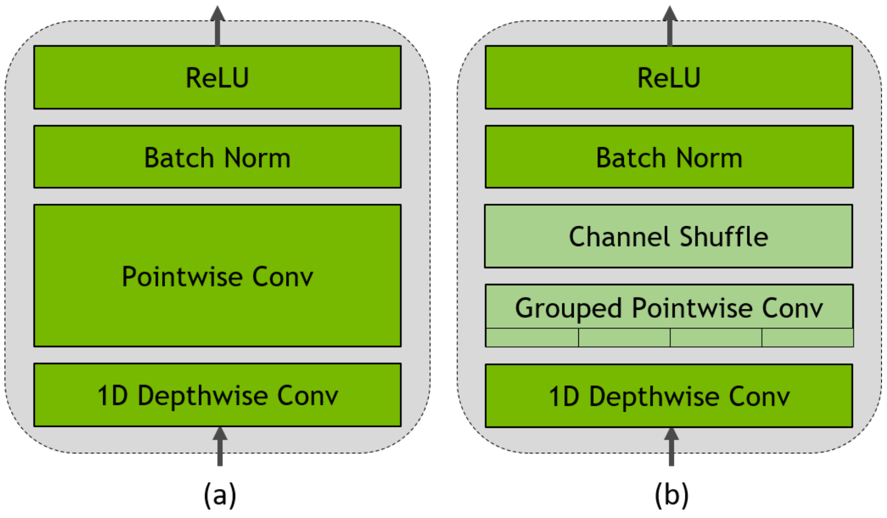


Figure 1.3: (a) Time-channel separable 1D convolutional module (b) Time-channel separable 1D convolutional module with groups and shuffle. Taken from Kriman et al. [2019].

1.3.3 Transformer

Transformer [Vaswani et al., 2017] has become a very well established architecture in Neural Machine Translation [Bojar et al., 2018, Barrault et al., 2019]. The main idea behind the architecture is to get rid of recurrence and convolutions and rather base the model solely on attention. As the attention mechanism is implemented as matrix multiplications, contemporary GPUs can better parallelize the computations leading to faster training.

A Transformer model is composed of a encoder and a decoder (see Figure 1.4). Both encoder and decoder are stacked identical layers. The encoder layer has two sub-layers: a multi-head self-attention layer and a fully connected feed-forward network. Decoder layer has an extra multi-head attention sub-layer, which performs attention over the output of the encoder. This additional layer is between the self-attention and feed-forward layers. Self-attention in the decoder is modified so that the auto-regressive property holds, i.e., the encoder cannot look to the right side ("future").

The advantage of self-attention is that it can access an arbitrary position in a constant number of sequentially executed operations while recurrent networks need $O(n)$ sequentially executed operations. If the sequence length n is less than the representation dimension d of the model, which is often the case (as the state-of-the-art machine translation models use sub-word units), the total computational complexity is lower than of the recurrent models. Another advantage presented by the authors is that self-attention leads to better interpretability of the models. They claim the individual attention heads seem to learn some specific functions that may be related to the syntactic and semantic structure of a sentence.

We employ this model in our enhanced ASR pipeline as a correction and language model and as a translation model in our SLT pipeline.

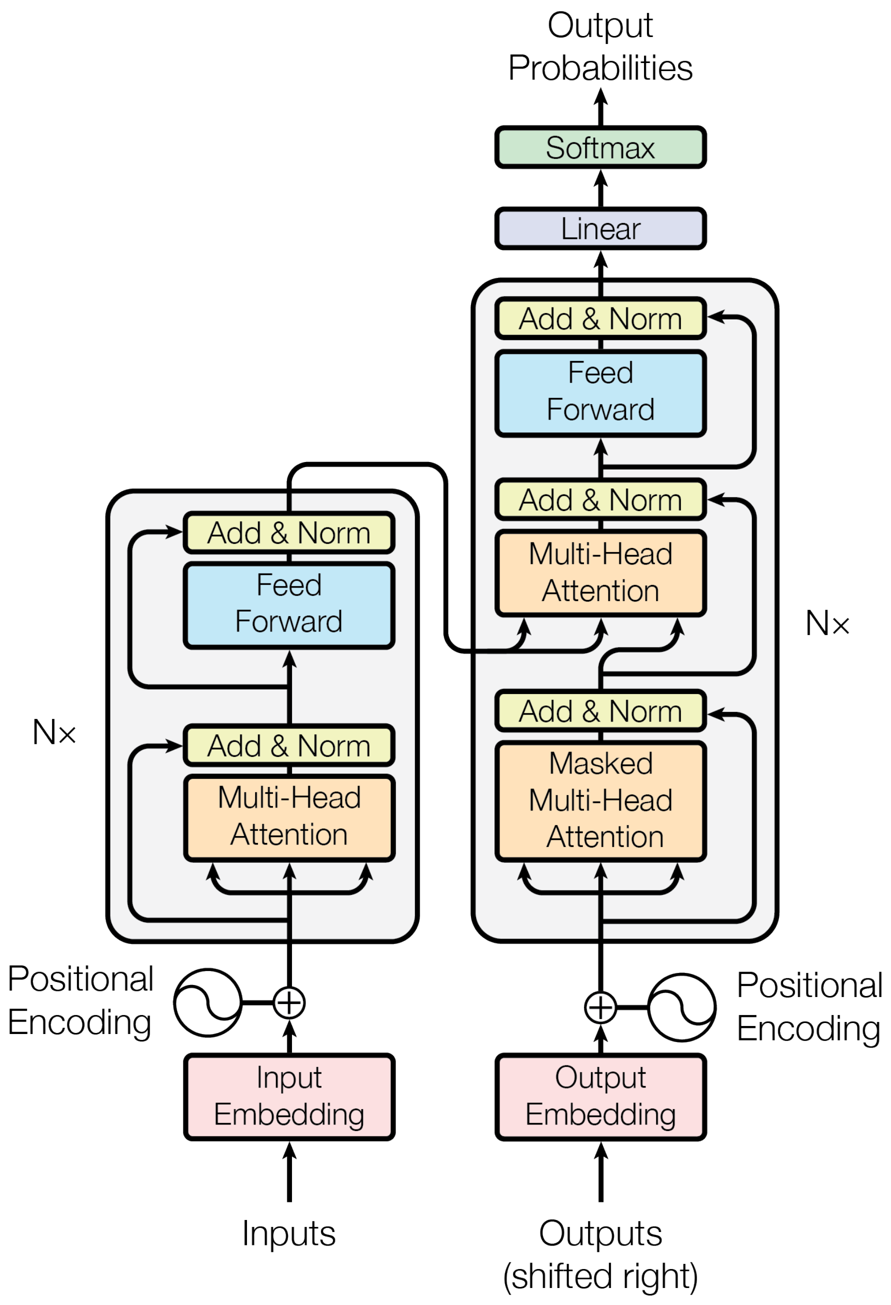


Figure 1.4: Transformer model architecture with detailed encoder (left) and decoder (right). Taken from Vaswani et al. [2017]

1.4 Data Representation

This section introduces the input and output data representations. The way how we encode and feed the neural network can significantly influence performance. In this work, we transcribe recordings, i.e., we work with voice and text data. First, we introduce MFCC — the voice presentation, and then we discuss text encoding.

1.4.1 MFCC

Mel frequency cepstral coefficients (MFCC) is the most commonly used representation of speech for ASR and SLT. This method exploits the way how the human auditory system perceives voice. The filter in the MFCC pipeline is linearly spaced for frequencies up to 1000 Hz and logarithmically above.

MFCC pipeline as described in Muda et al. [2010] and Kamath et al. [2019]:

1. **Pre-emphasis** Application of filter that emphasizes higher frequencies:

   $$Y[n] = X[n] - \alpha X[n-1] \tag{1.1}$$

   .

   The α coefficient is typically between 0.95 and 0.99. This filter amplifies the higher frequencies of the signal.

2. **Framing** Raw audio is segmented into small windows. The signal within each of the small windows can be then treated as stationary. Typically, the length of the window is about 20 ms, and windows have an overlap of 10 ms.

3. **Windowing** To avoid potential abrupt changes caused by framing, windowing is applied. Windowing is a multiplication of samples in a window with a scaling function. The most commonly used windowing in ASR is that of Hann and Hamming:

   $$w(n) = \sin^2\left(\frac{\pi n}{N-1}\right) \tag{1.2}$$

   $$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \tag{1.3}$$

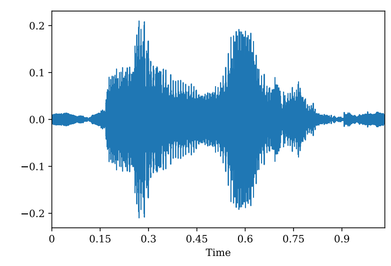
   where N is window length and $0 \leq n \leq N-1$.

4. **Fast Fourier Transform** FFT converts the one-dimensional signal from the time to the frequency domain.

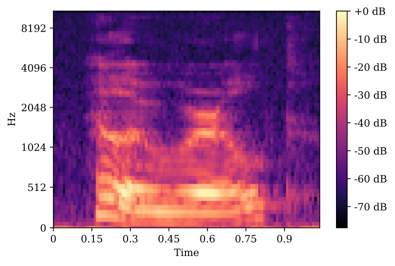
5. **Mel Filter Bank** The Mel Filter Bank is a set of filters that mimic the human auditory system. Usually, 40 filters are used. Each filter is of a triangular shape. Each filter produces the weighted sum of the spectral frequencies corresponding to each filter. These values map the input frequencies to the mel scale (a perceptual scale of pitches judged by listeners to be equal in distance from one another proposed by Stevens et al. [1937]).

6. **Discrete Cosine Transform** This process converts the log Mel spectrum into the time domain. The result is called the Mel Frequency Cepstrum Coefficient (the MFCC), and the set of coefficients is called acoustic vectors.

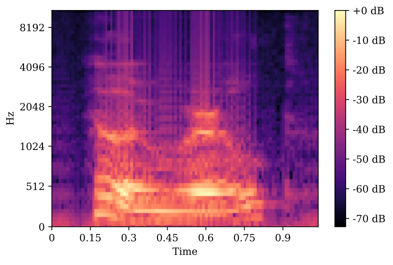
An example of a mel-spectrogram is in Figure 1.5.



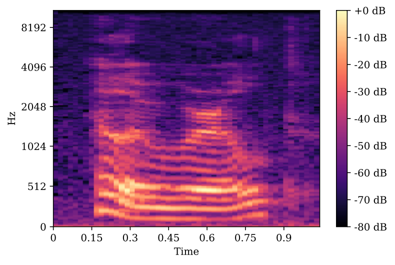
(a) Amplitude



(b) Mel filters: 64; window: 20 ms; overlap: 10 ms; window: hann. (Jasper defaults).



(c) Mel filters: 64; window: 20 ms; overlap: 10 ms; window: rectangular.



(d) Mel filters: 256; window: 50 ms; overlap: 0 ms; window: hann.

Figure 1.5: Amplitude and mel spectograms of "Hello World!".

1.4.2 Text Representation in NMT

There are many approaches to text representation in NMT. The first decision that has to be made is the size of the basic text unit. Text can be treated as a sequence of atomic:

- characters,
- words,
- sub-word units.

The first one is a very simple and straightforward approach. Character representation permits one to encode any word (of a given alphabet). Its downside is that it produces longer sequences compared with other methods. Less output classes lead to the reduction of computational complexity. However, the model needs to attend more positions, which substantially increases time complexity during decoding.

Word level representation, on the other hand, produces shorted sequences which may be better in some applications. Generally, though, neural machine translation is an open-vocabulary problem. From this point of wiev, word-level representation is undesirable, because it cannot handle unknown words. Several techniques have been proposed, such as NMT with a post-processing step [Luong et al., 2014, Luong and Manning, 2016].

The most versatile method seems to be sub-word representation. It addresses both problems, as it produces shorter sequences compared, and is also capable of handling unknown and rare words. The number of contemporary NMT systems that use sub-word level representation demonstrates its utility. The most prominent sub-word level tokenizers are BPEs [Sennrich et al., 2016] and subword regularization Kudo [2018]. Both methods are based on similar ideas — they produce more compact text representations. The former is based on the "merge" operation that joins the most frequent character sequences together, while the latter is based on unigram language model. A particular benefit of the subword regularization is that it is also able to produce different segmentation of the same text during the encoding.

In our work, we use the BPE implementation YouTokenToMe.[1] We chosen this particular implementation as it supports multithreading is considerably faster than other implementations and it offers Python as well as command-line interface. Further, it comes with BPE-dropout [Provilkov et al., 2019]. BPE-dropout is an enhancement of traditional BPE, which addresses the deterministic nature of the method. BPE-dropout randomly drops some merges from the BPE merge table during input segmentation, which results in slight variance of word segmentation. Introducing a noise to the data helps to regularize NMT model training.

Byte Pair Encoding

We would like to point out inconsistency of reporting BPE size in literature. Some authors use terms the "*BPE size*" and "*number of merge operations*" as synonyms, although the actual *BPE size* equals *number of merge operations* plus *the size of the alphabet*. In this work, we use the term "*BPE size*" to refer to the absolute vocabulary size — including the size of the alphabet.

[1] `https://github.com/VKCOM/YouTokenToMe`

1.5 Data Sets

We dedicate this section to data sets used for the training of models in our work. First, we introduce speech corpora LibriSpeech and Common Voice, and then we introduce the parallel corpus CzEng.

1.5.1 LibriSpeech

LibriSpeech [Panayotov et al., 2015] is a large corpus of read English speech. The corpus contains 1000 hours of transcribed speech based on audiobooks from project VoxForge[2].

The data set is structured into three parts that have approximately 100, 360, and 500 hours. Using a trained model on the Wall Street Journal corpus [Paul and Baker, 1992], authors divided the speakers by WER into two pools: “clean” and “other”. From the “clean” pool, 20 male and female speakers were randomly selected to development and test sets. The rest was assigned to 100 and 360 hours of “clean” sets. For the “other” pool (500 hours), authors selected more challenging data for development and test sets.

1.5.2 Common Voice

Common Voice [Ardila et al., 2019] is a multi-lingual, crowdfunded speech corpus. At the time of writing, 29 languages were available. English data set contained 1118 hours of validated, transcribed recordings. Unfortunately, the Czech data set was not available.

All utterances are collected and validated by volunteers. The data collection runs solely online through a web form. Speech utterances are stored in MPEG-3 format with a 48 kHz sampling rate. After at least two out of three volunteers up-votes an utterance, it is considered to be valid.

The number of clips is divided among the three datasets according to statistical power analyses. Given the total number of validated clips in a language, the number of clips in the test set is equal to the amount needed to achieve a confidence level of 99% with a margin of error of 1% relative to the number of clips in the training set. The same is true of the development set.

1.5.3 Large Corpus of Czech Parliament Plenary Hearings

Large Corpus of Czech Parliament Plenary Hearings [Kratochvíl et al., 2019] is a corpus of Czech transcribed speech. In our work, we use this dataset for the training of Czech ASR. The corpus has approximately 400 hours. As usual, the dataset contains training, evaluation, and test sets with no overlap. Training and development sets may have some common speakers. The test set should have no speaker overlap with the rest of the corpus.

A notable contrast with the previously mentioned speech corpora is the segmentation. The authors segmented the recordings into short sound clips (the longest one has 44s), disregarding the sentences.

[2] `http://www.voxforge.org/`

1.5.4 CzEng 1.7

CzEng 1.7 is a Czech-English parallel corpus containing about a greater billion words in each language. CzEng 1.7 is a filtered version of CzEng 1.6 [Bojar et al., 2016a]. The advantage of the CzEng corpus is its diverse set of origins such as subtitles, EU legislation, fiction, web pages, technical documents, or medical data.

Filtered CzEng

We use CzEng for training of ASR and SLT phoneme-to-grapheme translation models. To convert the dataset to graphemes, we utilize `phonemizer.`[3].

For this purpose, we removed all sentence pairs that "probably" do not occur in spoken language. We assume that "spoken utterances" will contain only the following character types: characters, numbers, apostrophes, punctuation, currency sign, dash, and quotation marks on the English side mark the sentence pair as a "probable" spoken utterance. Moreover, we filtered out too long and too short sentences (sentences must have at least two characters, at most 511 characters).

We intend to filter out sentences as for example:

E-006961/11 (PL) Marek Henryk Migalski (ECR) to the Commission (15 July 2011)

Such utterances do not have straightforward pronunciation, could break the `phonemizer`, and could degrade the transcript and translation quality.

[3]`https://github.com/bootphon/phonemizer`

1.6 Error Metrics

In this work, we use two error metrics, one for measuring the quality of speech recognition systems — word error rate (WER) — and a metric for evaluation of spoken language translation — BLEU.

1.6.1 Word Error Rate

Word error rate is one of the most commonly used metrics. This metric measures edit distance (insertions, deletions, and substitutions) between target and hypothesis:

$$WER = 100 \times \frac{I + D + S}{N} \tag{1.4}$$

where

- I is number of insertions,
- D is number of deletions,
- S is number of substitutions,
- N is number of words in target.

Character error rate is defined similarly where, instead of words, characters are considered.

1.6.2 BLEU

For the evaluation of translation tasks, we use the BLEU (Bilingual Evaluation Understudy) score introduced by Papineni et al. [2002]. This metric assigns scores in the interval $[0, 1]$, where 1 is the perfect score. The method counts occurrences of matching n-grams in the candidate and the reference (there can be more than one reference translations). It computes a modified n-gram precision (the number of occurrences of an n-gram in the candidate sentence must not exceed the maximum number of occurrences in any reference, otherwise it is clipped) and does a weighted geometric mean. In addition to the implicit penalization of excessive length, the metric introduces the brevity penalty.

Relying on a direct comparison of BLEU scores across papers was risky because its actual implementation could vary. We therefore use `SacreBLEU` proposed by Post [2018] in our work.

2. Automatic Speech Recognition

This chapter is devoted to automatic speech recognition (ASR). Throughout this chapter, we describe in detail our modifications to the process of training an ASR pipeline. One of the central challenges of this work is the scarcity of Czech training data. We propose a technique to overcome this problem.

This chapter is organized as follows: Section 2.1 presents and compares two techniques for overcoming the shortage of Czech ASR data. Section 2.2 describes transfer learning from graphemes to phonemes and the training of acoustic models for the final ASR/SLT pipeline.

2.1 Coarse-to-Fine Intermediate Step and Cross-Lingual ASR Transfer

In this section, we review techniques for overcoming training data shortage. Note, this section was submitted in the format of a paper to the Interspeech conference and it is now under review.

2.1.1 Introduction

Contemporary end-to-end, deep-learning automatic speech recognition systems achieved state-of-the-art results on many public speech corpora, see e.g. Han et al. [2019] on test-clean set from LibriSpeech.

To outperform traditional hybrid models, deep-learning ASR systems, however, must be trained on vast amounts of training data, on the order of a thousand of hours. Currently, there is only a limited number of public datasets that meet these quantity criteria. The variety of covered languages is also minimal. In fact, most of these large datasets contain only English. Although new speech datasets are continually emerging, producing them is a tedious and expensive task.

Another downside of new end-to-end speech recognition systems is their requirement of an extensive computation on many GPUs, taking several days to converge, see e.g., Karita et al. [2019].

These obstacles are often mitigated with the technique of transfer learning [Tan et al., 2018] when a trained model or a model part is reused in a more or less related task. Furthermore, it became customary to publish checkpoints alongside with the neural network implementations and there emerge repositories with pre-trained neural networks such as *TensorFlow Hub*[1] or *PyTorch Hub*.[2] This gives us an opportunity to use pre-trained models, but similarly, most of the published checkpoints are trained for English speech.

In our work, we propose a cross-lingual coarse-to-fine intermediate step and experiment with transfer learning [Tan et al., 2018], i.e., the reuse of pre-trained models for other tasks. Specifically, we reuse the available English ASR checkpoint of QuartzNet [Kriman et al., 2019] and train it to recognize Czech speech instead.

[1]`https://tfhub.dev/`

[2]`https://pytorch.org/hub/`

This section is organized as follows. In Section 2.1.2, we give an overview of related work. In Section 2.1.3 we describe the used models and data. Our proposed method is described in Section 2.1.4, and the results are presented and discussed in Section 2.1.5. Finally, in Section 2.1.6 we summarize the work

2.1.2 Related Work

Transfer learning [Tan et al., 2018] is an established method in machine learning because many tasks do not have enough training data available, or they are too computationally demanding. In transfer learning, the model of interest is trained with the help of a more or less related "parent" task, reusing its data, fully or partially trained model, or its parts.

Transfer learning is step by step becoming popular in various areas of NLP. For example, transferring some of the parameters from parent models of high-resource languages to low-resource ones seems very helpful in machine translation [Zoph et al., 2016, Kocmi and Bojar, 2018].

Transfer learning in end-to-end ASR is studied by Kunze et al. [2017]. They show that (partial) cross-lingual model adaptation is sufficient for obtaining good results. Their method exploits the layering of the architecture. In essence, they take an English model and freeze weights in the upper part of the network (closer to the input). Then they adapt the lower part for German speech recognition yielding very good results while reducing training time and the amount of needed transcribed German speech data.

Other works concerning end-to-end ASR are Tong et al. [2018] and Kim and Seltzer [2018]. The former proposes unified IPA-based phoneme vocabulary while the latter suggests a universal character set. The first demonstrates that the model with such alphabet is robust to multilingual setup and transfer to other languages is possible. The latter proposes language-specific gating enabling language switching that can increase the network's power.

Multilingual transfer learning in ASR is studied by Cho et al. [2018]. First, they jointly train one model (encoder and decoder) on ten languages (approximately 600 hours in total). Second, they adapt the model for a particular target language (4 languages, not included in the previous 10, with 40 to 60 hours of training data). They show that adapting both encoder and decoder, boosts performance in terms of character error rate.

Coarse-to-fine processing [Raphael, 2001] has a long history in NLP. It is best known in the parsing domain, originally applied for the surface syntax [Charniak et al., 2006] and recently for neural-based semantic parsing [Dong and Lapata, 2018]. The idea is to train a system on a simpler version of the task first and then gradually refine the task up to the desired complexity. With neural networks, coarse-to-fine training can lead to better internal representation, as e.g., Zhang et al. [2018] observe for neural machine translation.

The term coarse-to-fine is also used in the context of hyperparameter optimization, see e.g., Moshkelgosha et al. [2017] or the corresponding DataCamp class,[3] to cut down the space of possible hyperparameter settings quickly.

Our work is novel and differs from the abovementioned in two ways: First, we reuse existing models and checkpoints to improve the speed and accuracy of an unrelated

[3]`https://campus.datacamp.com/courses/hyperparameter-tuning-in-python/informed-search?ex=1`

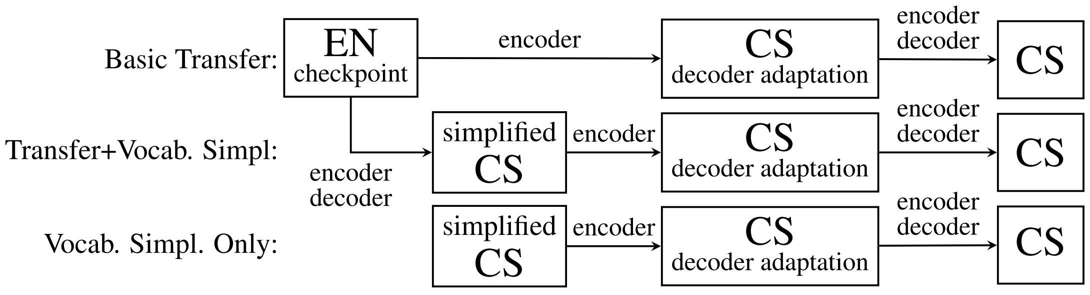


Figure 2.1: Examined setups of transfer learning. The labels on the arrows indicate which model parts are transferred, i.e., used to initialize the subsequent model. No parameter freezing is involved except for the encoder weights in the "CS decoder adaptation" phase.

language. Second, in the coarse-to-fine step method, we simplify (instead of unifying) Czech character set in order to improve cross-lingual transfer, but also to enhance monolingual training significantly.

2.1.3 Data and Models Used

Pre-Trained English ASR. As the parent English model we use the checkpoint available at the *NVIDIA GPU Cloud*.[4] It is trained for 100 epochs with batch size 512 on 8 NVIDIA V100 GPUs and achieves 3.98 % WER on LibriSpeech [Panayotov et al., 2015] test-clean.

During the experiments, the model configuration provided by the NeMo authors is used with minor changes (we used 1000 warm-up steps and learning rate 10^{-3} for decoder adaptation). We note that we use $O1$ optimization setting. Which primarily means mixed-precision training (weights are stored in single precision, gradient updates are computed in double precision). We perform training on 10 NVIDIA GeForce GTX 1080 Ti GPUs with 11 GB VRAM.

Czech Speech Data. In our experiments, we use Large Corpus of Czech Parliament Plenary Hearings [Kratochvíl et al., 2019]. At the time of writing, it is the most extensive available speech corpus for the Czech language, consisting of approximately 400 hours.

The corpus includes two held out sets: the development set extracted from the training data and reflecting the distribution of speakers and topics, and the test set which comes from a different period of hearings. We choose the latter for our experiments because we prefer the more realistic setting with a lower chance of speaker and topic overlap.

A baseline, end-to-end Jasper model, trained on this corpus for 70 epochs, has an accuracy of 14.24 % WER on the test set.

2.1.4 Examined Configurations

Figure 2.1 presents the examined setups. In all cases, we aim at the best possible Czech ASR, disregarding the performance of the model in the original English task. The baseline (not in the diagram) is to train the network from scratch on the whole

[4]`https://ngc.nvidia.com/catalog/models/nvidia:quartznet15x5`

Czech dataset, converting the speech signal directly to Czech graphemes, i.e., words in fully correct orthography, except punctuation and casing which are missing in both the training and evaluation data.

Basic Transfer Learning

The first method is very similar to Kunze et al. [2017]. We use the English checkpoint with the (English) WER of 3.98 % on LibriSpeech test-clean, and continue the training on Czech data.

Czech language uses an extended Latin alphabet, with diacritic marks (acute, caron, and ring) added to some letters. This extended alphabet has 42 letters, including the digraph “ch”. Ignoring this digraph (it is always written using the letters “c” and “h”), we arrive at 41 letters. Only 26 of them are known to the initial English decoder.

To handle this difference, we use a rapid decoder adaptation. For the first 1500 steps, we keep the encoder frozen and train the decoder only (randomly initialized; Glorot uniform).

Subsequently, we unfreeze the encoder and train the whole network on the Czech dataset.

Transfer Learning with Vocabulary Simplification

In this experiment, we try to make the adaptation easier by first keeping the original English alphabet and extending it to the full Czech alphabet only once it is trained.

To coerce Czech into the English alphabet, it is sufficient to strip diacritics (e.g. convert “čárka” to “carka”). This simplification is quite common in Internet communication but it always conflates two sounds ([ts] written as “c” and [tʃ] written as “č”) or their duration ([aː] for “á” and [a] for “a”).

In this experiment, we first initialize both encoder and decoder weights from the English checkpoint (English and simplified Czech vocabularies are identical so the decoder dimensions match), and we train the network on the simplified Czech dataset for 40 thousand steps.

The rest (adaptation and training on the full Czech alphabet) is the same as in Section 2.1.4. Overall, this can be seen as a simple version of coarse-to-fine training where a single intermediate model is constructed with a reduced output alphabet.

Vocabulary Simplification Only

In the last experiment, we disregard the English pre-trained model and use only our vocabulary simplification trick. We first train the randomly initialized model on Czech without diacritics (26 letters) for 38 thousand steps. We then adapt the pre-trained, simplified Czech model to the full Czech alphabet with diacritics (41 letters), again via the short decoder adaptation. Note that the original decoder for simplified Czech is discarded and trained from random initialization in this adaptation phase.

Experiment	**Simplified**	**Adaptation phase**	**Full training**
Baseline CS	-	-	20.19
EN $\rightarrow$ CS	-	97.35	19.06
EN $\rightarrow$ sim. CS $\rightarrow$ CS	17.11	22.01	16.88
sim. CS $\rightarrow$ CS	20.56	24.59	16.57

Table 2.1: Results in % of word error rate on the Czech (CS) test set. “Simplified” column reflects WER after the training on simplified dataset (both training and test data without accents). “Adapt.” column contains WER immediately after the decoder adaptation phase to the full Czech alphabet including accents. Finally, “Full” column contains performance on the test set with accents after the full training.

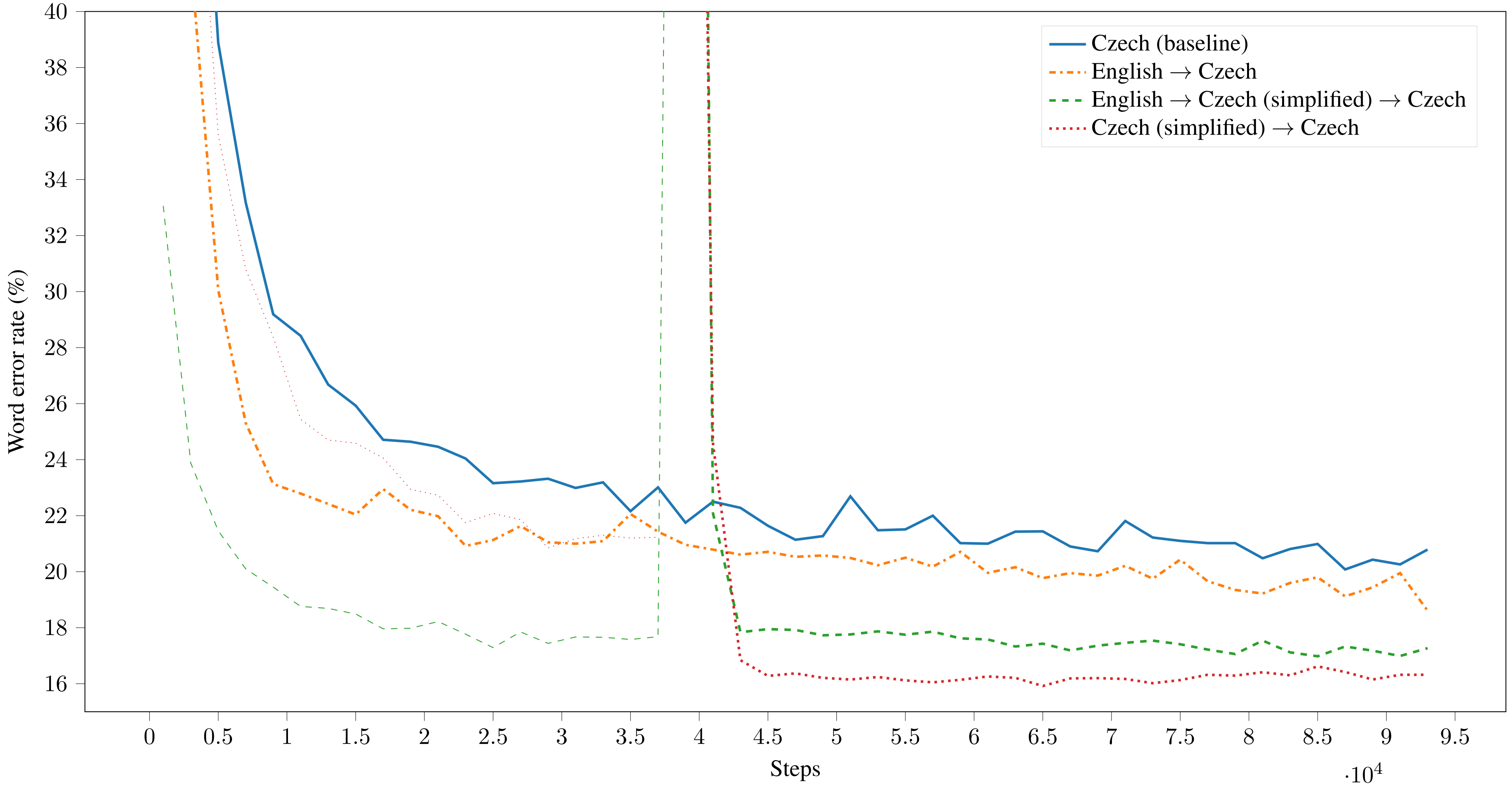


Figure 2.2: Evaluation on test set during training. Note, that WER curves for experiments with simplified vocabulary (thin lines) are not directly comparable with other curves until step 40,000 as the test set is on a different (simplified) vocabulary. 10,000 steps takes approximately 5 hours of time.

2.1.5 Results and Discussion

Table 2.1 presents our final WER scores and Figure 2.2 shows their development through the training. For simplicity, we use greedy decoding and no language model. We do not use a separate development set. We simply take the model from the last reached training iteration.[5]

Transfer across Unrelated Languages

We observe that initialization with an unrelated language helps to speed-up training. This is best apparent by comparing the first 40k steps of the learning curves for "English $\rightarrow$ Czech simplified" and "Czech (simplified) $\rightarrow$ Czech" in Figure 2.2, thin line. Here the target alphabet was simplified in both cases but the weights initialized from English allow much faster decrease of WER. The benefit is clear also for the full alphabet (baseline vs. "English $\rightarrow$ Czech") where the baseline has a consistently higher WER.

The training off the English parent is actually so fast that WER for the simplified alphabet (thin dashed green line) drops under 30 % within the first 2000 steps (1 hour of training). This can be particularly useful if the final task does not require the lowest possible WER, such as sound segmentation.

While Basic transfer ("English $\rightarrow$ Czech") boosts the convergence throughout the training, its final performance is only 1 to 2 % points of WER better than the baseline, see the plot or compare 20.19 with 19.06 in Table 2.1. The intermediate vocabulary simplification is more important and allow a further decrease of 2.2 % WER absolute (19.06 vs. 16.88) when transferring from English.

Transfer across Target Vocabularies

In the course of two experiment runs, we altered the target vocabulary: the training starts with simplified Czech, and after about 40,000 steps, we switch to the full target vocabulary. This sudden change can be seen as spikes in Figure 2.2. Note that WER curves prior to the peak use the simplified Czech reference (the same test set, but with stripped diacritics), so they are not directly comparable to the rest.

The intermediate simplified vocabulary always brings a considerable improvement. In essence, the final WER is lower by 2.18 (16.88 vs. 19.06 in Table 2.1) for the models transferred from English and by 3.62 (16.57 vs. 20.19) for Czech-only runs. One possible reason for this improvement is the "easier" intermediate task of simpler Czech. Note that the exact difficulty is hard to compare as the target alphabet is smaller, but more spelling ambiguities may arise. This intermediate task thus seems to help the network to find a better-generalizing region in the parameter space. Another possible reason that this sudden change and reset of the last few layers allows the model to reassess and escape a local optimum in which the "English $\rightarrow$ Czech" setup could be trapped.

2.1.6 Conclusion and Future Work

We presented our experiments with transfer learning for automated speech recognition between unrelated languages. In all our experiments, we outperformed the baseline in

[5]Little signs of overfitting are apparent for the "Simplified CS $\rightarrow$ CS" setup so that an earlier iteration might have worked better, but we do not have another test set with unseen speakers to validate it.

Set	**Greedy**	**Beam search with LM**
Development	5.19	4.91
Test	7.64	6.42

Table 2.2: Our best Czech model. Results in % WER measured after fixing errors in development and test set transcriptions.

terms of speed of convergence and accuracy.

We gain a substantial speed-up when training Czech ASR while reusing weights from a pre-trained English ASR model. The final word error rate improves over the baseline only marginally in this basic transfer learning setup.

We are able to achieve a substantial improvement in WER by introducing an intermediate step in the style of coarse-to-fine training, first training the models to produce Czech without accents, and then refining the model to the full Czech. This coarse-to-fine training is most successful within a single language: Our final model for Czech is better by over 3.5 WER absolute over the baseline, reaching WER of 16.57%.

We see further potential in the coarse-to-fine training. We would like to explore this area more thoroughly, e.g., by introducing multiple simplification stages or testing the technique on more languages.

2.1.7 Note on the Results

After reviewing the Large Corpus of Czech Parliament Plenary Hearings Kratochvíl et al. [2019], we found out that the transcripts contained systematic errors (the "SIL" tag for silence was accidentally left in at the beginnings and ends of some transcripts). This resulted in a systematic mismatch of the system output and the reference transcript during testing.

Unfortunately, we made this finding too late to be able to redo all the experiments. Table 2.2 compares the WER scores of the best model on the original and fixed reference transcripts (without any model retraining). We also add scores with beam search and a 5-gram KenLM [Heafield, 2011] language model, confirming the improvement we expected. The final score is thus 4.91 on the development and 6.42 % WER on the test set, ten points better than the scores reported in Table 2.1.

2.2 Speech Recognition to Phonemes

In this section, we build ASR systems for English and Czech with phonemes as target vocabulary. During the training, we build upon our findings from the previous section.

This section is organized as follows: First, in Section 2.2.1 we review related work and discuss the motivation for the transition from graphemes to phonemes as target vocabulary. Next, in Section 2.2.2 we present the experiment set-up. Section 2.2.3 gives details about training. Finally, we evaluate results in Section 2.2.4.

2.2.1 Related Work

Phonemes are well-established modeling units in speech recognition. From the beginnings of the ASR technology in the 1950s, researchers employed phonemes [Juang and Rabiner, 2005].

For the GMM-HMM models, a state typically represents part of a phone [Yu and Deng, 2016]. A *phone* is a linguistic unit representing a sound regardless of the meaning in a particular language. On the other hand, we have *phonemes* that are also linguistic units representing a sound. However, opposed to the phones, if a phoneme is switched for another one, it could change the meaning of a word in a particular language. A common trick for improving the quality is to combine phones into *diphones* (two consecutive phones) or *triphones* (three successive phones) [Kamath et al., 2019].

Riley and Ljolje [1992] perform a comparison of different linguistic units for sound representation. Namely, they compare phonemic (using phones), phonetic (using phonemes), and multiple phonetic realizations with associated likelihoods. All experiments use the GMM-HMM ASR model. Using the phonemic representation, the authors were able to achieve 93.4 % word accuracy on DARPA Resource Management Task. Using the most probable phonetic spelling, the authors improved the result to 94.1 %. Authors also propose using more phonetic realizations for each word, allowing different pronunciations. These are obtained automatically from the TIMIT phonetic database. In their best setup, which achieves an accuracy of 96.3 %, they report the average of 17 representations per word.

An important piece of work popularizing neural networks in ASR is that of Waibel et al. [1989]. The work proposes to use a time-delayed neural network to model acoustic-phonetic features and to model the temporal relationship between them. The authors demonstrate that the proposed NN can learn shift-invariant internal abstraction of speech and use this to make a robust final decision. The authors choose phonemes as the modeling unit.

More recent studies that still use hybrid models (mainly, HMM and DNN) employ multi-task learning to improve phone recognition task significantly.

For example, Seltzer and Droppo [2013] improve the primary task of predicting acoustic states (61 phonemes, three states each) using a different secondary assignment. They propose three additional problems: (1) Phone Label Task, where the system predicts phone labels for each acoustic state. (2) State Context Task, in which the model predicts the previous and following acoustic state. (3) Phone Context Task, where the model predicts the previous and next phone.

Another example of multi-task learning in ASR is Chen et al. [2014]. The authors use a DNN for the triphone recognition task. To improve the performance in this task, they suggest using the additional task of trigrapheme recognition, which they claim to be highly related. In their setup, both problems share the internal representation (the hidden layers of DNN). The authors successfully demonstrate that their contribution outperforms other single task methods and even the ROVER system [Fiscus, 1997] that combines results from the triphone and trigrapheme tasks.

Finally, though not directly from the ASR field, an excellent utilization for phoneme ASR in speech translation is suggested by Salesky et al. [2019]. For their end-to-end speech translation pipeline, they first obtain phoneme alignment using the DNN-HMM system. They average feature vectors with the same phoneme for consecutive frames. Phonemes then serve as input features for end-to-end speech translation. XXX Salesky pouziva Kaldi a phonemy vnima ako ”hustejsiu” reprezentaciu zvuku, takze cele jej SLT je tym padom E2E.

2.2.2 Experiment Set-Up

Based on the abovementioned review of related work, we design our experiment. The main goal of our work is to build spoken language *translation* system. This has a significant impact on our decision-making about the rough outline.

We target our ASR to phonemes rather than graphemes. We presume they are more suitable linguistic modeling units for the recognition task, while they keep the problem similar to the grapheme recognition task (mainly supported by Riley and Ljolje [1992], Juang and Rabiner [2005], Chen et al. [2014]). This is perhaps more true for Czech where the phonetical and graphical transcriptions are relatively alike. We specifically decided for IPA phonemes and not for other phonetic units like phones. We believe that the IPA are adequate and we can quickly get phonetical transcription for most languages using the `phonemizer`[6] tool.

We find the end-to-end system architecture for ASR promising for the future. Therefore, unlike the DNN-HMM architecture used by the Salesky et al. [2019], we bet on E2E architecture. Notably, for English, we employ the bigger Jasper architecture [Li et al., 2019a], and for Czech, we utilize the smaller QuartzNet architecture [Kriman et al., 2019]. Our decision to use different architectures for English and Czech is motivated by the volume of the training data and the number of trainable parameters of the respective models. For English, we have almost 2000 hours of transcribed speech and 333 million of parameters in the Jasper model while for Czech only about 400 hours of transcribed speech and 18.9 million of parameters in the QuartzNet model.

We take advantage of the fact that the grapheme and phoneme recognition tasks are similar. We employ transfer learning from pre-trained grapheme models. Inspired by our previously proposed “Adaptation method” (see Section 2.1.4), we first strip the “grapheme” decoder. We replace the stripped grapheme decoder with a new decoder with correct number of output classes to account number of phonemes. This new decoder is then randomly initialized (using Glorot uniform). During the adaptation phase we adapt the model (only a randomly initialized decoder). After the adaptation, we train the whole model (encoder and decoder together).

2.2.3 Training

In this section, we review the training of the experiment, as described in Section 2.2.2. Note that we have different training procedures for English and Czech. The training

[6] `https://github.com/bootphon/phonemizer`

scheme is common to both languages: first, the Adaptation phase, and second, the Full training.

We use mixed-precision training, which means that the weights are stored in `float16` (making the model smaller), and weight updates are computed in `float32` precision (reducing the quality loss).

When it comes to phoneme segmentation, one technical question arises. Both trained architectures output character labels, but some phonemes are multi-character (for example “aː”, “dʒ”, “aɪ”, “aɪə” or “aɪɚ”,). We already dealt with this problem in the previous section due to the Czech digraph “ch”. We decided to disregard this character in the ASR to Czech graphemes as it can be easily rewritten using “c” and “h”. Here the problem is more frequent. Many phonemes in both languages are multi-character (Czech has 13 and English 27). Examples of words with different multi-character phonemes are in Table 2.3 on page 31. From the examples, we see that some of the multi-character phonemes consist even of other single-character ones. Hence, disregarding this issue might influence the model performance. Bearing this in mind, we conduct two additional experiments. The first experiment obeys the phoneme segmentation, and the second breaks all phonemes to single characters. As the English model has much greater hardware requirements, we train only Czech variants and assume the results will carry over to English.

We first attempt to follow the phoneme segmentation given by the `phonemizer`. `phonemizer` can output phonetic transcriptions including separators between phonemes and words. We use “|” as the phoneme separator (see Table 2.3). XXX Note, this separator is used by the data preparation layer in the acoustic model. The layer splits the input phoneme string on the separator symbols, and each substring is translated to one phoneme label. Hence, the neural network is unaware of the separator symbol. Specifically, the acoustic model output does not contain any separators.

In the second attempt, we decompose all multi-character phonemes (e.g., “dʒ” is decomposed to “d” and “ʒ”). Note, some of the single-character phonemes are encoded in `UTF-8` as multi-character using combining letters. To circumvent this, we additionally break single-character phonemes that consist of combining letters and we replace these special letters with a simple Latin letter substitute. The substitute letters I, L, T, R were chosen so that they do not occur in the respective language. So, for example the Czech word “případně” with the phonetical transcription “pr̝̊iːpadɲe” is rewritten to “prTRiːpadɲe”.

Czech ASR For training and testing, we use the “phonemized” corpus of the Czech Parliament Plenary Hearings. As the dataset is five times smaller than English corpora, we utilize the smaller QuartzNet.

We train off the best model from the previous experiment described in Section 2.1. This model yields WER of 4.91 % on the development and 6.42 % the test set on the Parliament corpus using beam search with decoding.

The Adaptation Step takes 2000 steps. As the model’s memory footprint is smaller during this phase, we increase the batch size to 256. Two thousand steps are warm-up, the maximal learning rate is 4×10^{-3}.

The full training takes 30000 steps. The model memory requirements increase, therefore we reduce the batch size to 32. We also reduce the learning rate to 10^{-3}.

Both experiments with phonemes alphabets (the broken and non-broken one) use the same configuration.

English ASR For the training and evaluation of English ASR, we translate to phonemes the corpora LibriSpeech and Mozilla Common Voice (again using the `phonemizer` tool). We use both (shuffled) datasets for the Adaptation and Full training phases.

The training is executed on 10 NVIDIA GTX 1080 Ti GPUs with 11 GB VRAM.

We train off the Jasper ASR model as available online.[7] This model was trained on the LibriSpeech, Mozilla Common Voice, WSJ, Fisher, and Switchboard corpora. The model yields 3.69 % on the test-clean, and 10.49 % on the test-other using greedy decoding on the LibriSpeech.

The Adaptation phase takes 2000 steps. As the model's memory footprint is smaller during this phase, we increase the batch size to 64 (global batch size is 640). One thousand steps are warm-up; the maximal learning rate is 4×10^{-3}.

The Full training takes ten epochs. The model memory requirements increase, therefore we reduce the batch size to 16 (global batch size is 160). We also reduce the learning rate to 10^{-3}.

2.2.4 Evaluation

In this section, we review the course of training and evaluate the models' performance.

Note, the results are not directly comparable with graphemes. The mapping to phonemes is not a 1-1. For example, the two words "I am" map in American English to only one "phoneme word" [aɪæm]. For this reason we carefully distinguish between the standard word error rate and phoneme word error rate (PWER) which measures the same Levenshtein distance on the sequence of phoneme tokens.

Czech The course of the Czech training is displayed in Figure 2.3 and the final evaluation is in Table 2.4.

As expected, a rapid PWER fall at the beginning of the training (see Figure 2.4) proves that the grapheme and phoneme recognition tasks are indeed similar because the pre-trained weights adapt so quickly and well to the new target units.

We made a surprising observation: beam search decoding is always worse than greedy decoding. We tested the beam sizes of power two from 1 to 64. We use two different beam search implementations: TensorFlow[8] XXX for beam search without LM rescoring and Baidu's CTC decoder[9] with KenLM [Heafield, 2011] language model.

Using XXX beam search without LM rescoring (TensorFlow), we got the best result 9.14 % PWER, about 0.2 % percent point higher than with the greedy search for multi-character ASR. For single-character ASR, the difference is even more significant: 9.09 % for greedy and 9.53 % PWER for beam search.

We also tried adding a language model to the second CTC decoder.[10] Using KenLM, we trained a 3-gram language model on phonemized Czech part of the CzEng. For both single- and multi-character phonemes, the language model helps. The result for the single-character setup is a bit better.

[7]`https://ngc.nvidia.com/catalog/models/nvidia:multidataset_jasper10x5dr`

[8]`https://www.tensorflow.org/`

[9]`https://github.com/PaddlePaddle/DeepSpeech`

[10]When setting LM weight to higher than 0, we found that this implementation does not support labels containing more that one character, which causes a problem to the several multi-character phonemes in the phonetic alphabet.

Phonemes	Examples	
“i”, “ə” and “iə”	s|ɪ|ɹ|**i**|ə|s	v|ɛ|ɹ|**iə**|s
“aɪ”, “ə” and “aɪə”	dʒ|**aɪ**|ə|n|t	k|w|**aɪə**|t

Table 2.3: Examples of multi-character phonemes consisting of other single- and multi-character phonemes. “|” is the segmentation mark. These segmentation marks are present in ASR training data, but not in the language model training data.

XXX Phoneme segmentation discrepancy in trainig data We suspect the difference stems from our trick (substituting multi-character phonemes with a new single character — see Section 2.2.3) to overcome Baidu’s CTC decoder limitation. Using this trick, we substitute multi-character phonemes with a unique symbol in transcripts (used for the acoustic model training) and language model training data.

The language model training data (unlike the transcripts) does not contain phoneme boundaries (i.e., there is no phoneme separator token “|” present). Some multi-character phonemes consists of two other valid single-character (or even multi-character) ones (see Table 2.3). Thus, during the substitution of multi-character phonemes, some neighboring phonemes might be considered as one phoneme and incorrectly substituted. Note, the ASR model is trained on data with known phoneme segmentation (given by the `phonemizer`). Hence, during the decoding, the acoustic model can sometimes attribute a higher probability to two phonemes instead of one, while the language model gives to these phonemes lower score (because of the training data mismatch of the language and ASR model). An illustration of this problem is in Table 2.3. For example, the phonetic transcriptions of “serious” and “various” are sɪɹiəs and vɛɹiəs. They both share common suffix — ɹiəs. But, when we consider the segmentation produced by the `phonemizer`, “various” has one phoneme iə and “serious” has two phonemes — i and ə.

To test this assumption, we train a 3-gram language models (for single- and multi-character ASR) on ASR training data (notably a considerably smaller amount of data then CzEng) that include phoneme segmentation (e.g., “v|ɛ|ɹ|**iə**|s”). Note, we do not re-phonemize CzEng, as phonemization is time and power-consuming process. Using these data that include segmentation, we correctly substitute multi-character phonemes. After applying the substitution of the multi-character phonemes, we remove the phoneme separators (as the CTC decoder’s queries to the LM are without separators).

The best results are 8.30 and 7.74 % PWER for multi- and single-character ASR respectively. These results are consistent with the greedy and beam search without LM decoding. Therefore, we conclude that using native phoneme segmentation is slightly better. Hence, for English, we use multi-character phonemes.

English The course of the English training was carried out on two development sets and it is pictured in Figure 2.4. The final results on the development and test sets are in Table 2.5.

We again test beam search with and without a language model. We train a 3-gram LM and use ASR training data, as these contain separated phonemes. The obtained results could be, therefore, better if we took more training data for the language model and bigger n-grams.

Again, as the observed performance in Table 2.5 suggests, beam search decoding performs worse than greedy decoding. Adding a language model to re-score beams

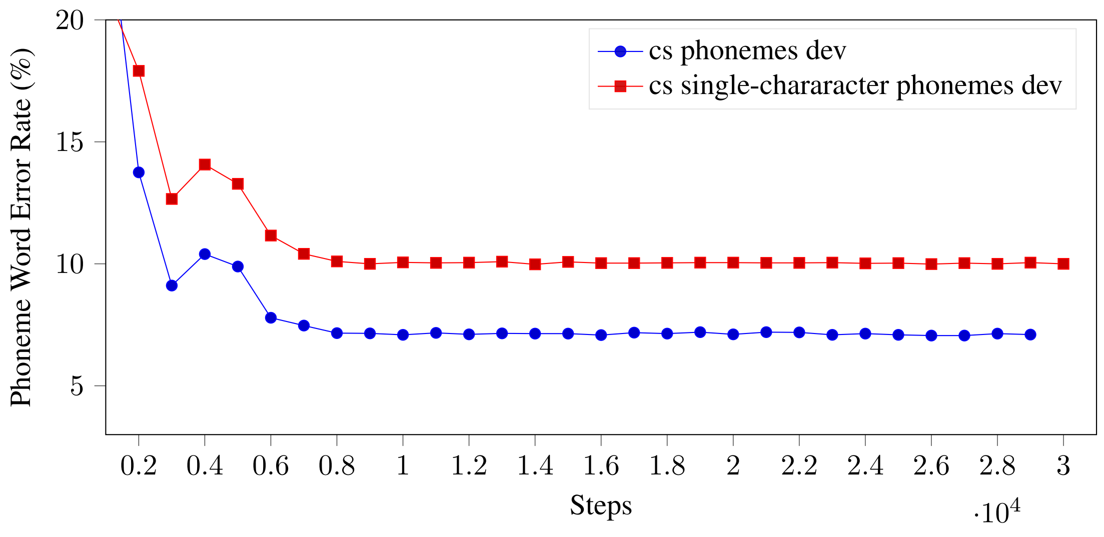


Figure 2.3: Learning curves on phonemized Czech Parliament Hearings development set.

Type	Method	Adaptation phase	Full training		
			greedy	beam s.	beam s. with LM
dev	multi-char. phonemes	13.75	7.09	-	-
	single-char. phonemes	17.91	10.04	-	-
test	multi-char. phonemes	-	8.94	9.14	8.30
	single-char. phonemes	-	9.09	9.53	7.84

Table 2.4: Results in % of *Phoneme* Word Error Rate (PWER). The language model is trained on CzEng. Note, PWER is not directly comparable to WER. The column "Adaptation phase" represents model performance after the adaptation of the decoder to new modeling units but before further training (i.e., the encoder is frozen and only decoder is trained).

helps to reduce PWER. There is a slight reduction in the Libri Speech test-clean. Test-other gets better about 3 % PWER absolute. For the Common Voice test set, the PWER reduction is considerable — almost 40 % relative (6.46 vs. 10.21) compared to the greedy decoding. As a possible explanation why LM helps Common Voice so much is that nearly two-thirds of the LM training data come from this data set and there is a significant overlap of sentences in the training data and the test set.

2.2.5 Conclusion and Future Work

We proposed an alternative text representation for ASR — phonemes — and trained models for Czech and English language. Further, we examined two different approaches for phoneme encoding: respecting multi-character phonemes and single-character ones. We concluded that using native, multi-character phonemes is slightly better. We also successfully trained language models for use with phonetic transcriptions.

The question whether the phoneme-based ASR works better than grapheme-based ASR, cannot be answered simply. The mapping between graphemes and phonemes is not straightforward. Therefore, we leave it to the next chapter.

In the future, we would like to examine coarse-to-fine methods for improving

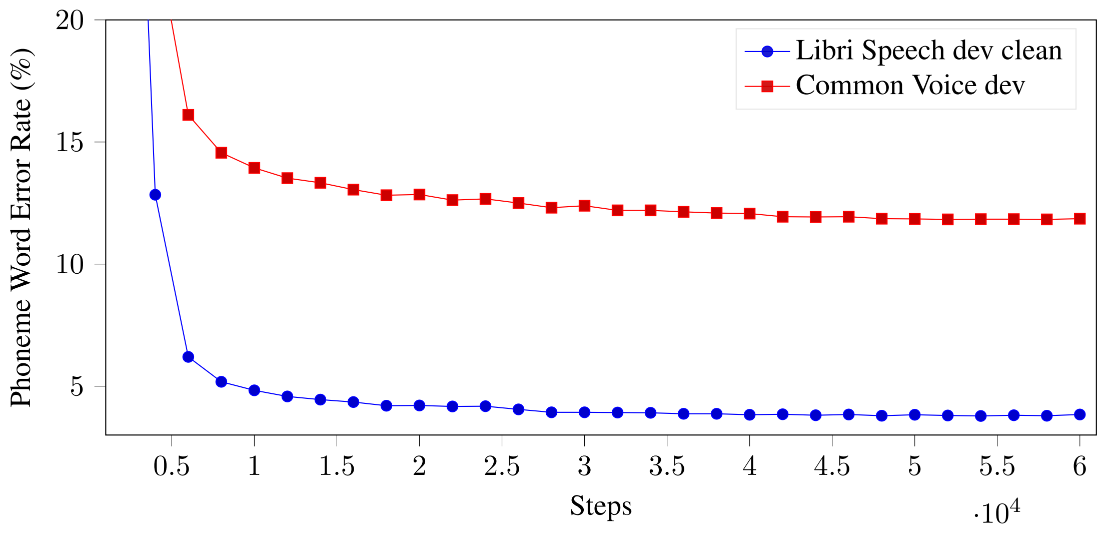


Figure 2.4: Learning curves on phonemized Libri Speech `dev-clean` and Common Voice `dev`.

Type	Corpus	Adaptation phase	Full training greedy	beam s.	beam s. with LM
dev	Libri Speech Clean	46.07	3.84	-	-
	Common Voice	54.69	11.86	-	-
test	Libri Speech Clean	-	4.18	4.48	3.58
	Libri Speech Other	-	11.48	11.67	8.57
	Common Voice	-	10.21	10.47	6.46

Table 2.5: Results in % of *Phoneme* Word Error Rate (PWER). The language model is trained on phonemized ASR training data. Note, PWER is not directly comparable to WER. XXX The column "Adaptation phase" represents model performance after the adaptation of the decoder to new modeling units but before further training (i.e., the encoder is frozen and only decoder is trained).

phoneme-based ASR. We believe that similarly to grapheme-based ASR as observed in Section 2.1, this could reduce the training time and lower the final PWER.

Another technique, multi-task learning, proposed and applied in HMM-based ASR by Chen et al. [2014], could bring a further performance boost.

3. Enhanced ASR

In this chapter, we describe and build an enhanced ASR model. We propose to split a conventional end-to-end ASR into to two successive models:

1. an "acoustic" model that outputs phonemes instead of graphemes,
2. a "translation" model that consumes the phonemes and translates them to graphemes. Note that the translation does not involve any other language, see Chapter 4 for true translation.

An illustration of the proposed enhanced ASR pipeline is in Figure 3.1.

The added hope is that the translation model after the "acoustic" model will not only "blindly" translate phonemes into graphemes, but also correct errors. Errors can occur, for example, due to adverse conditions during voice recording (e.g., background noise), speaker's dialect, or pronunciation errors. Some of these errors may be not apparent a person, as we are naturally able to communicate in a noisy environment. The motivation for the introduction of such a translation step into our pipeline is that such a model better "understands the language" and can take an extended context into account compared to the rather convolutional Jasper model. Furthermore, we can utilize other non-speech corpora, e.g., available monolingual data to train this part of our pipeline.

We decided to use phonemes as an intermediate representation. We believe that conventional grapheme representation is too complicated and constrained for some languages with complicated rules of mapping speech to a transcript. This issue becomes immense when dealing with dialects and non-native speakers. The translation model in the second step thus may be a better fit for this transformation.

Step by step, we describe the selection of hyperparameters and training of the translation model for the enhanced ASR pipeline. First, we discuss and experiment with source encoding, and afterwards, we train the model.

The chapter is organized as follows: we first review related work in Section 3.1. In Section 3.3, we experiment with various approaches to text encoding. In Section 3.4 we present the main objective of this chapter — monolingual translation model for ASR.

3.1 Related Work

In this section, we review related work. We examine ASR-related work in Section 3.1.1, and in Section 3.1.2 we take a closer look at text encoding.

3.1.1 ASR

We already reviewed the usage of phonemes in ASR in Section 2.2.1. At this point, we further expand on the corresponding work.

One of the main challenges of this chapter is to build a phoneme-to-grapheme (P2G) translation model. In most studies, P2G translation is utilized for enhancing ASR. More precisely, for out-of-vocabulary (OOV) words, the ASR system outputs phonemes instead of graphemes. P2G then tries to find a proper orthographic representation. Examples of studies employing P2G in this manner are Decadt et al. [2001], Horndasch et al. [2006], Basson and Davel [2013].

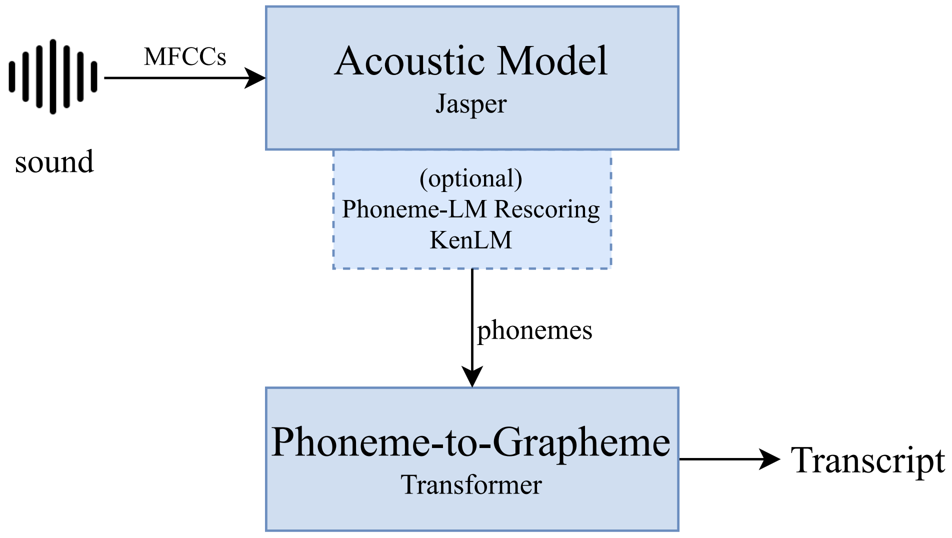


Figure 3.1: Enhanced ASR pipeline. The input sound is first transcribed to phonemes using Jasper/QuartzNet “acoustic” model. Phonemes are fed to the “translation” model. Translation model not only translates, but also fixes some errors in the ASR output and produces orthographic transcriptions.

One of the first attempts on P2G translation is Reddy and Robinson [1968]. The authors propose to use a tree-structure mechanism to keep track of the possibilities at various stages combined with phoneme-to-word dictionary and the structure of the English language. To speed up the translation process, the authors divide the dictionary into ten sections determined by parts of speech. Authors also consider erroneous input. They propose to have more dictionary entries (for example, to account for dialects) and human aid.

Another study in P2G translations is conducted Decadt et al. [2001]. They apply P2G to enhance the readability of out-of-vocabulary (OOV) output in speech recognition. In their setup, ASR outputs standard (orthographic) text for known words. For OOVs, phonemes are outputted. Because the phonemes are hard to read for most users, the authors propose to translate phonemes using memory-based learning [Daelemans et al., 2004]. Specifically, they offer two approaches. First, they use a similarity metric to find closest examples in the lexicon (features, i.e. phonemes, are weighted using gain ratio) and extrapolate the result from them. Second, they propose to use the IGTREE algorithm [Daelemans et al., 1997]. The authors, surprisingly, report that the final word error rate in their setup (Dutch ASR) is higher than the baseline.

At the same time, the output should be better readable for a person. They report that 41 % words are transcribed with at most one error, and 62 % have only two errors. Furthermore, most of the incorrectly transcribed words do not exist in Dutch.

Horndasch et al. [2006] introduce a data-driven approach called MASSIVE. Their main objective is to find appropriate orthographic representations for dictated Internet search queries. Their system iteratively refines sequences of symbol pairs in different alphabets. In the first step, they find the best phoneme-grapheme alignment using the expectation-maximization algorithm [Dempster et al., 1977]. In the second step, they join neighboring symbols together to account for the insertions. Finally, n-gram probabilities of symbol pairs are learned. During the inference, the input string is split into individual symbols. For each symbol all possible symbol pairs are generated. According to the beam width, the best sequences are taken. On the German CELEX lexical database, they obtain 96.1 % word accuracy. For English CMU, they obtain an accuracy of 87.2 % for 10-fold cross-validation.

3.1.2 Text Encoding

In this section, we give a brief overview of work related to text encoding. A high-level review of text representation in NMT can be found in Section 1.4.2. Here we provide a small survey of text encoding techniques aimed at various techniques for enhancing translation quality.

Note that some authors use terms "*BPE size*" and "*number of merge operations*" as synonyms. The actual *BPE size*, however equals *the number of merge operations* plus *characters*. In most cases, the number of characters is small relative to the merge-operations. Then the discrepancy is negligible. In our work, we use the term "*BPE size*" as the total number of unique entries in the BPE dictionary.

After some early experiments, e.g. Bojar et al. [2016b], the first systematic attempt to study the impact of BPE vocabulary size in Neural Machine Translation was made by Denkowski and Neubig [2017]. Specifically, they compare full-word systems with 16k and 32k BPEs. In their setups, they use *shared vocabularies* — BPE learned from the concatenation of the source and target data sets. They conclude that using BPE is better than using full-word vocabularies. For BPE, they recommend using a larger vocabulary over a smaller one in high-resource setups (in their case, over 1M parallel sentences). Reviewing their results, we observe only a little performance degradation using 16k (smaller) BPE in high-resource setups. More precisely, the loss is 0.4 BLEU in the DE-EN translation task and none for other configurations. A slightly better performance in low-resource tasks means 0.3 and 0.4 BLEU for the EN-FR and the CS-EN, respectively.

Cherry et al. [2018] went in a different direction — towards character encoding. In their study, the authors compare BPE and character encoding in combination with LSTM NMT. They claim that artificial representations such as BPE are sub-optimal, leading to, e.g. (linguistically) improbable word fragmentations. Although these representations outperform BPE, the authors acknowledge the problem of much higher computational requirements for both training and inference.

A more in-depth study of different setups (architectures, joint vs. separate BPE, languages) and the impact of vocabulary sizes on NMT performance is offered by Ding et al. [2019]. The authors review several configurations and a broad range of BPE sizes ranging from character-level to 32K. They show that using an appropriate setting can help gain 3 to 4 BLEU points. Most experiments with smaller vocabularies (sizes up to 4K) performed better for the low-resource configuration. For a high-resource setting, the larger BPE sizes are however better. The authors also study joint and separate BPE. They conclude that the difference is negligible.

Gupta et al. [2019] study character-based and BPE NMT with Transformer under various conditions. They conclude that BPE with 30K vocabulary should be the standard choice in high resource settings. They also experiment with noisy data: when training on clean data, BPE performs slightly worse than character-level encoding. However, when training on corrupted data, BPE with a large vocabulary (30000) performed better than character-level or BPE with a smaller vocabulary. We speculate that larger units help in recovering from noise. In the low-resource scenario, character level models perform better. In the high-resource scenario, however, large BPE models outperform other settings. The only exception is the WMT Biomedical test set, which contains a large proportion of technical terms.

For our specific usecase of converting phonemes to graphemes, Hrinchuk et al. [2019] use BERT [Devlin et al., 2018] original 30K WordPieces vocabulary and do not examine other sizes or other training data.

3.2 Experiment Motivation and Outline

As reviewed in the previous section, we observe that the utilization of intermediate phonetic transcription is an established method in ASR. Therefore, we propose an SLT pipeline which consists of an acoustic model followed by a translation model. As the intermediate representation unit, we propose phonemes (see Figure 3.1 on page 35).

Unlike the most studies reviewed in Section 3.1.1, we propose to use Transformer architecture for phoneme-to-grapheme translation. We believe that Transformer is the best option for these tasks. Transformer has shown its potential in many NLP tasks. Most importantly, we consider its ability to learn the structure of a sentence, see e.g. Pham et al. [2019]. We are convinced that this could help to reduce errors in transcripts. Our other motivation to apply this architecture is its task versatility. Just by changing training data, we can immediately train a spoken language translation system, i.e., translation of phoneme in a source language to graphemes in a target language, see Chapter 4.

We describe and evaluate the training of the Transformer phoneme-to-grapheme translation in Section 3.4.

Our survey in Section 3.1.2 demonstrates an essential role of text encoding on the translation model performance. Hence, preceding the model training, we first study the impact of tokenizer selection on P2G translation in Section 3.3.

3.3 Word Encoding Selection

In this section, we review the alternatives for the input and output tokenization. Because of the limited computing resources, we experiment only on English and assume the results will carry over to Czech.

Throughout this experiment, we use Byte Pair Encoding for the tokenization of the source and target sentences. We rule out word-level representation as it creates models with a worse performance (in terms of speed and quality) [Denkowski and Neubig, 2017]. Also, the word-level representation cannot deal with out-of-vocabulary items.

We decided to use separate source and target vocabularies. We are motivated by the findings of Ding et al. [2019]. In general, they did not observe the difference when using shared and separate vocabularies. They encourage practitioners to consider their particular use case. In our task, the input and output alphabets — phonemes and graphemes — are different. In this experiment, we choose 8k BPE for target tokenization. We follow the authors of the NeMo toolkit.[1] They claim this BPE size should help to lower the memory footprint and hence increase throughput leading to the faster convergence. We use phonemized CzEng corpus as training data.

As we previously discussed, the selection of training data and the size of vocabulary for target BPE is relatively straightforward. On the other hand, considering the source may be corrupted as the ASR system produces it in our setup, there arise two questions:

- is better to use smaller or larger vocabulary size?
- which training data to use to train the tokenizer (uncorrupted data translated from CzEng using `phonemizer`, data obtained from ASR or mixture of both)?

3.3.1 Experiment Outline

Our task — translation from phonemes to graphemes in the same language — differs from the situations described in previous work. Hence there is no clear answer to our previously stated questions.

To resolve these questions, we conduct a series of experiments. We train 16 Transformer models, each with different source vocabulary (sizes with a step of multiple of four: character-level, 128, 512, 2k, 8k, and 32k. Each BPE size except for character-level is trained on clean, corrupted, and mixed data (the procedure for generating corrupted data is described in Section 5.2). The target vocabulary remains the same for all configurations — 8k trained on clean graphemes, phonemized filtered CzEng 1.7. Afterwards, we evaluate their performance on "corrupted" development sets that were obtained in Section 5.2.

Considering the time and hardware complexity of training Transformer `big` configuration, we choose the `base` setting for these experiments. We believe the behavior of the model will still be reasonable alike.

3.3.2 Data Preparation

The BPE vocabulary of the *clean* setup uses clean filtered CzEng. The *corrupted* setup utilizes data obtained from acoustic models. We describe the procedure for creating these data in Section 5.2 on page 60. We have approximately 7M corrupted sentences. For *mixed data* setup, we take a random subset of 7M from clean filtered CzEng. We do not take the whole CzEng as it has 57M sentence pairs.

As training data for the Transformer models, we use corrupted data from our ASR ensemble setup. We selected two development sets: the `dev-clean` set from LibriSpeech and the `dev` set from Common Voice. The reason is that LibriSpeech contains longer utterances than Common Voice, but on the other hand, the former has lower WER than the latter. It is also worth noting that LibriSpeech `dev-clean`'s utterances are twice that long on average (107 characters LibriSpeech versus 52 characters Common Voice).

[1] `https://nvidia.github.io/NeMo/nlp/neural_machine_translation.html`

Size	**Clean**	**Corrupt**	**Mixed**
character	7.55	-	-
128	7.38	7.32	7.40
512	7.21	7.27	7.19
2k	7.12	7.20	7.22
8k	7.10	7.05	7.10
32k	**6.98**	**7.03**	**6.93**

Table 3.1: Results in % of word error rate on the Common Voice `dev` set. Best results in each column in bold.

Size	**Clean**	**Corrupt**	**Mixed**
character	5.82	-	-
128	6.03	5.69	6.69
512	5.54	5.50	5.49
2k	5.48	5.27	5.46
8k	5.44	5.39	5.47
32k	**5.18**	**5.24**	**5.25**

Table 3.2: Results in % of word error rate on the LibriSpeech `dev-clean`. Best results in each column in bold.

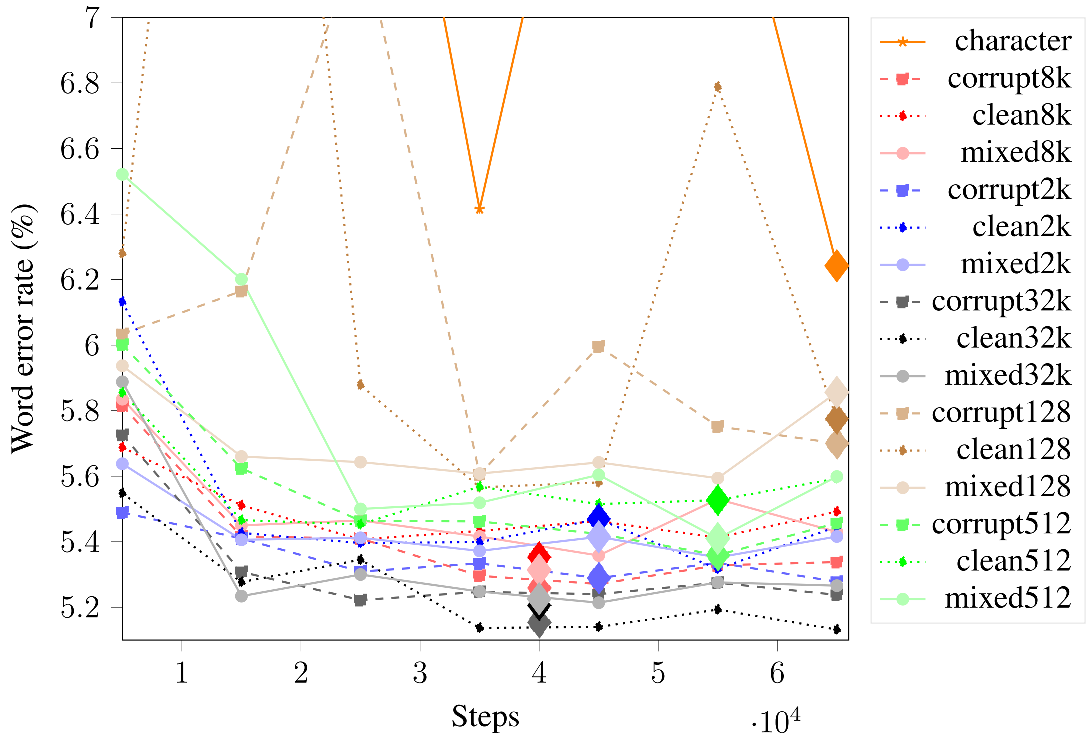


Figure 3.2: Each model is evaluated on LibriSpeech corrupted `dev-clean` (see Section 5.2) every 5000 steps. Bigger diamond marks shows where each trained model reached 10th epoch.

3.3.3 Training

As mentioned previously, we use the Transformer `base` configuration. We set the maximum sequence length to 1024 because of character-level, 128, and 512 BPE configurations. We train all models for 70000 steps on one GPU using the same batch size for all configurations: 12000 tokens.

An overview of runs is pictured in Figures 3.2 and 3.3. The final results on development sets of all experiments are in Tables 3.1 and 3.2.

3.3.4 Results and Analysis

The final results of all experiments are in Tables 3.1 and 3.2 for development sets and in Tables 3.3 to 3.5 for the test sets.

A graphical comparison is in Figures 3.4 to 3.6.

First of all, we can observe a drastic reduction of WER for the Common Voice `test` set compared to the LibriSpeech `test-other` relative to the acoustic model performance. The acoustic model has similar PWER scores on both test sets (see Table 2.5 on page 33).

We take a closer look at the training data. The filtered version of Common Voice has 611990 recordings, but only 81334 unique transcripts. This means that the same text has been recorded multiple times by many speakers. Common Voice "corrupted" has 3262524 unique ASR-provided transcriptions (meaning that a sentence has a unique error) and 80710 different true transcripts.

On the other hand, LibriSpeech has 281241 recordings and 281071 unique transcripts. Hence, there are almost none different recordings for the same text. "Corrupted" LibriSpeech has 3844983 unique erroneous ASR transcripts with 280850 different valid transcripts. Hence, on average, each Common Voice sentence has seven different recordings and 40 individual faulty ASR transcriptions. LibriSpeech, on the contrary, does not have more recordings per text and has about 13 ASR-corrupted transcripts per one original text. Hence, we are strongly convinced that the trained Transformer models are over-fitting the Common Voice dataset.

BPE Size Character-level encoding seems to be the worst or second-worst possible representation. For the Common Voice `test` set, it scores almost one percentage point of WER more compared to the best result (5.53 vs. 4.55). Also, all other vocabulary sizes performed almost half a percentage point better. For both LibriSpeech test sets, it performed a bit better than BPE 128.

Generally, Figures 3.4 to 3.6 suggest a clear trend: the larger the vocabulary, the lower word error rate. Among the different BPE sizes, we can recognize that the 32k vocabulary size has systematically the best results on all test sets.

Finally, we consider the following: a model can better learn from larger vocabulary sizes. First, a model does not have to learn low-level orthography extensively. Rather than memorizing the order of characters (or other smaller units), it can focus on the whole sentence and how individual words interact. Second, a larger model can detect errors because of anomalies in input encoding. Larger vocabularies produce a shorter representation. A corrupt word is more likely to be broken down to smaller pieces. When a model detects such a situation, it can, for example, identify the right target word based on the context, rather than the suspicious word. Such an anomaly will most likely not occur in text encoded with small BPE.

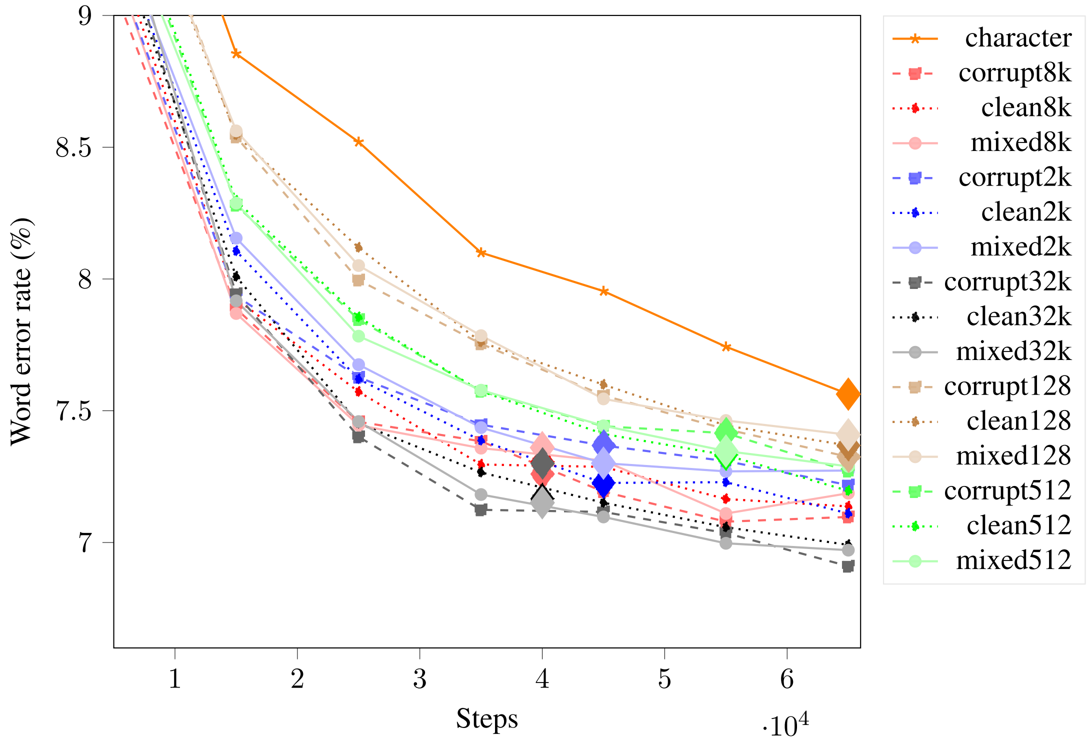


Figure 3.3: Each model is evaluated on Common Voice corrupted `dev` set (see Section 5.2) every 5000 steps. Bigger diamond marks shows where each trained model reached 10th epoch.

Size	**Clean**	**Corrupt**	**Mixed**
character	5.53	-	-
128	5.06	4.95	5.05
512	5.05	4.79	4.87
2k	4.86	5.09	4.97
8k	4.92	4.99	4.96
32k	**4.81**	**4.65**	**4.55**

Table 3.3: Results in % of word error rate on the Common Voice `test` set. Best results in each column in bold.

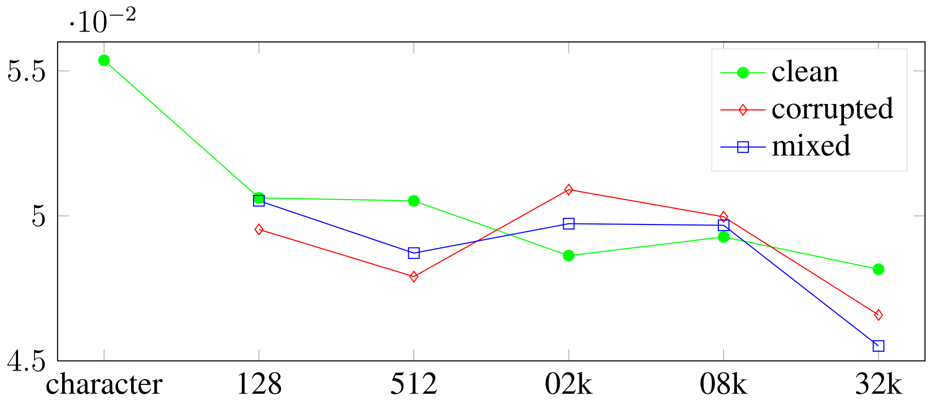


Figure 3.4: Results in % of word error rate on the Common Voice `test` set.

Size	Clean	Corrupt	Mixed
character	5.64	-	-
128	5.93	5.97	6.04
512	5.40	5.48	5.64
2k	5.34	5.30	5.34
8k	5.30	5.28	5.34
32k	**5.19**	**5.25**	**5.18**

Table 3.4: Results in % of word error rate on the LibriSpeech `test-clean`. Best results in each column in bold.

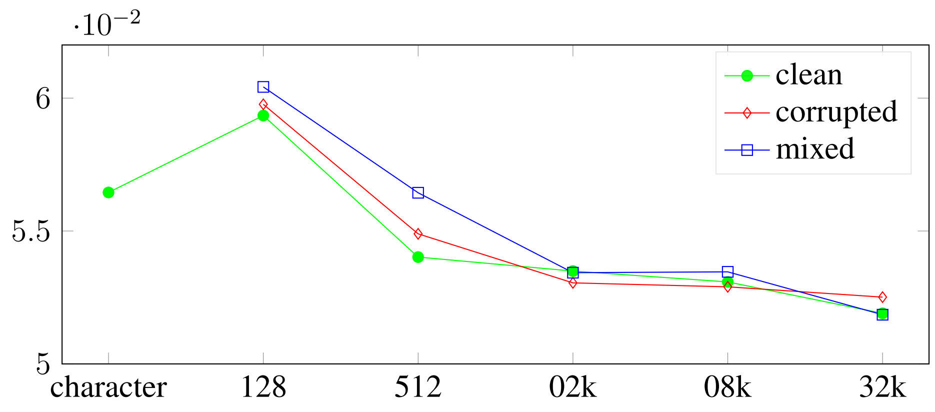


Figure 3.5: Results in % of word error rate on the LibriSpeech `test-clean`.

Size	Clean	Corrupt	Mixed
character	11.79	-	-
128	11.98	11.79	12.54
512	11.45	11.59	11.74
2k	11.60	11.45	11.47
8k	11.69	11.56	11.57
32k	**11.36**	**11.43**	**11.37**

Table 3.5: Results in % of word error rate on the LibriSpeech `test-other`. Best results in each column in bold.

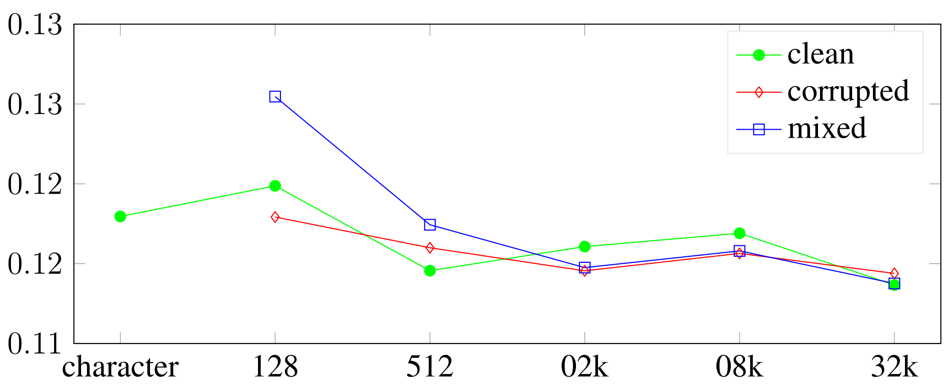


Figure 3.6: Results in % of word error rate on the LibriSpeech `test-other`.

Source of BPE Training Data For the LibriSpeech test sets, we observe almost no differences. Only the "corrupted" configuration has slightly less performance.

For Common Voice, we can witness some variation in performance. The best seems to be the "mixed" configuration. Somewhat worse is the "corrupted" and the worst is the "clean" version. In this case, we think the "mixed" is best as it has frequent enough "corrupted" words. This enables a model to learn to translate these corrupted words to correct ones. Also, it knows enough other words, so it can adequately work with correct phonemes.

Therefore, we conclude that the source of training data for BPE has almost no impact on the final result. One should consider sweeping various options for a more specific usecase (e.g., ASR for dictated speech in specific area).

3.3.5 Conclusion

We carried out an extensive study on the impact of BPE vocabulary size and BPE training data sources. Based on the empirical evidence, we assume it is better to use larger vocabularies. This is consistent with Gupta et al. [2019]: in high-resource setting (as ours: we train on 7M sentence pairs) when trained on corrupted data, the larger vocabularies are better.

In general, we do not see much difference between clean, corrupt, and mixed setting. In specific usecase (e.g., dictated speech), the selection of the training data source can influence the performance. In this case, one should consider to sweep various settings.

Therefore, in further experiments and setups, we will use 32k BPE vocabulary trained on clean training data.

3.4 English and Czech Enhanced ASR

In this section, we describe the training of the translation model for the proposed enhanced ASR pipeline (see Figure 3.1). We train both English and Czech "translation" models.

3.4.1 Experiment Outline

We seek a model for translating transcripts written in phonemes into graphemes in the same language. As previously discussed, we decided to use Transformer architecture. Translating phonemes into graphemes is a somewhat nontraditional task for the Transformer model. Therefore, we experiment with both variants of the architecture — "base" and "big".

In Section 3.3, we studied appropriate tokenizer. We concluded the larger the BPE vocabulary, the better performance the model has. We obtained the best results with 32k BPE. We also discussed the source of BPE training data (note, not the Transformer model training data). We settled that there is no essential difference between clean, corrupted, and mixed setup in general. Keeping this in mind, we use clean phonemized CzEng data for the BPE vocabulary training in this experiment.

First, we are interested in model XXX (note, the Transformer model, not the BPE encoding for which we decided to use clean data in all our experiments) performance when trained on clean data. In this chapter, we use clean phonemized CzEng. In Chapter 5 we further study fine-tuning of models for corrupted ASR data.

As already commented, this task is unusual for Transformer. Therefore, we try different beam sizes during the inference. We believe it could help the model to find better transcriptions by considering less likely labels.

3.4.2 Training Configuration

We tried to follow training tips for Transformer by Popel and Bojar [2018]. We found that these suggestions do not work well with the NeMo toolkit. We tried many hyper-parameter settings. We found the following to work best for purposes of this experiment:

For Transformer "base":

GPU(s): one GPU with 11 GB VRAM suffices

batch size: 6000

learning rate: 0.04

warmup steps: 8000

max steps: 600000 with manual abortion

For Transformer "big":

GPU(s): 6 GPUs with 16 GB VRAM

batch size: 6000

learning rate: 0.02

warmup steps: 16000

max steps: 600000 with manual abortion

We apply the BPE dropout [Provilkov et al., 2019] as a further regularization technique. We use the dropout probability suggested by the authors — 0.1.

Overview of training captured by evaluation on phonemized `newstest2015` is in Figures 3.7 and 3.8.

Beam size	**WER "base"**	**WER "big"**
1	10.53	9.56
4	10.41	9.59
16	10.47	9.60
Baseline	7.64	

Table 3.6: Results in % of word error rate on the Czech Parliament test set. The baseline is a standalone ASR, a QuartzNet network, transcribing directly to the graphemes. We take this model from Chapter 2.

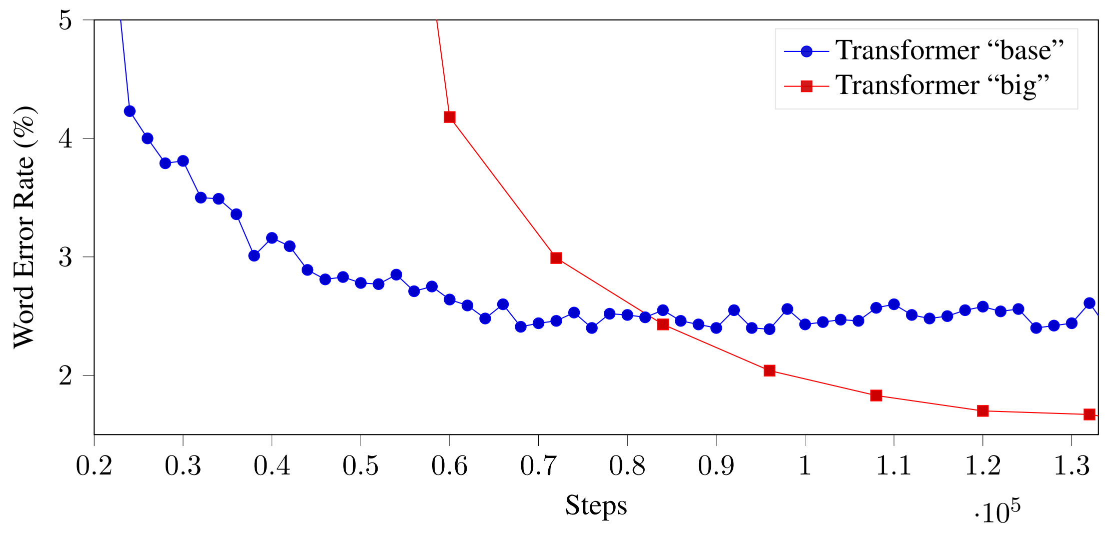


Figure 3.7: Evaluation on `newstest2015` during the training of the Czech P2G translation model. Phonemized Czech CzEng side serves as source and original Czech side as target.

Corpus	**Beam size**	**WER "base"**	**WER "big"**
	1	10.14	9.77
Common Voice	4	10.02	9.72
	16	10.01	9.75
	1	5.46	4.89
LibriSpeech `test-clean`	4	5.36	4.8
	16	5.37	4.85
Jasper baseline		3.86	
	1	12.11	11.74
LibriSpeech `test-other`	4	11.99	11.67
	16	12.00	11.67
Jasper baseline		11.95	

Table 3.7: Results in % of word error rate on the Common Voice `test` and the LibriSpeech `test-clean` and `test-other`. Jasper baseline taken from original Jasper paper [Li et al., 2019a].

3.4.3 Performance Evaluation

In this section we evaluate Czech and English phoneme-to-grapheme translation models trained on clean phonemized data. The performance of the respective models on test sets is in Tables 3.6 and 3.7.

Beam Size The gap between the individual beam sizes is negligible. It seems that the models are confident about the predictions at each step and do not need to consider other alternatives. Given the "clean" phonemized data, this is not something unexpected. Later, when the model will be exposed to "corrupted" data, there could be some change in the behavior.

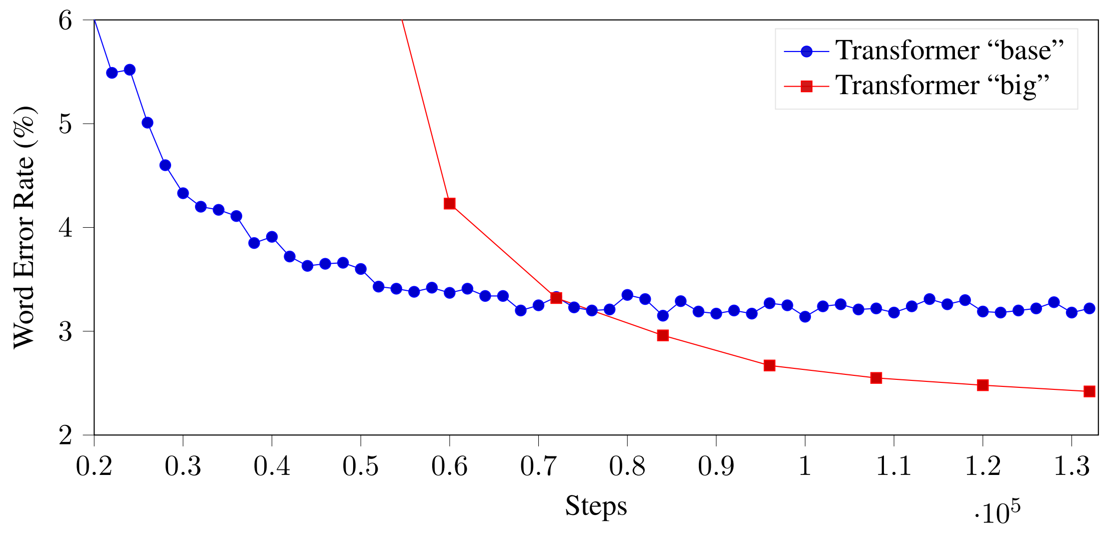


Figure 3.8: Evaluation on `newstest2015` during the training of the English P2G translation model. Phonemized English CzEng side serves as source and original English side as target.

Transformer "Base" vs. "Big" From the results in both languages, we see that the bigger model outperforms the smaller. The margin is rather narrow — at most, 0.5 % WER. We assume that later when we introduce errors to training data, the performance gap between the variants gets higher.

Czech Enhanced ASR Table 3.6 compares performance in terms of word error rate on the Czech Parliament corpus test set.

When compared to the baseline (our best model from Chapter 2), a plain ASR model, we see significant performance degradation. Baseline scores 7.64, and the best result for the Czech Enhanced ASR is 9.56.

First, we manually inspect obtained transcriptions. We see the translation model is inconsistent with outputting numbers — it appears that the model arbitrarily decides to write a number as a sequence of digits or a sequence of words. Essentially, this is not an error. We see that the predicted number is always correct. But the WER metric is not able to distinguish it and counts it as an error.

Further, we observe the model to "try hard" to translate the input "as is". The model is trained to transcribe correct transcriptions blindly. The model applies this behavior also to the erroneous data, producing incorrect transcriptions.

Second, we must consider the quality of phonetic transcriptions. These are obtained from Section 2.2. The evaluation metric for these models is the phoneme word error rate. The PWER and the WER are not directly comparable. The model we used to obtain phonetic transcriptions has PWER 8.94 %. In general, we suppose the PWER to be a bit worse. Some orthographically separate words (like "I am", or prepositions) become one phoneme word. Making a mistake in such a phoneme word can lead to a greater error rate. But there can be other linguistic phenomena. If we, for a brief moment, considered WER and PWER comparable, we observe the slightly better performance of the grapheme model. We believe there is still room for training a better acoustic model. As observed in Chapter 2, transfer learning is better than training from scratch but might lead to sub-optimal results when compared with the proposed coarse-to-fine technique.

English Enhanced ASR Table 3.7 compares performance in terms of word error rate on three test sets — Common Voice test, LibriSpeech `test-clean`, and `test-other`. Again, same as in Czech, we apply three different beam sizes (1, 4, and 16).

On the LibriSpeech `test-clean`, the performance is slightly inferior compared to the baseline. On the test other, the WER is comparable. It appears the acoustic model is the bottleneck.

Similarly to the Czech, the model disregards orthography and tries to match grapheme to phoneme transcriptions. The overall quality hence follows the quality of the acoustic model.

3.4.4 Conclusion

We proposed and trained an enhanced ASR system. We experimented with the two Transformer variants — "base" and "big" — and we also applied different beam sizes during the inference.

We found, the bigger models to perform better than the smaller ones. Further experiments on ASR and SLT will be therefore using the bigger architecture.

Using different beam sizes did not lead to much different results. Nevertheless, there is still prospect of some difference when trained on corrupted data.

The trained models did yet not outperform the baseline. But their performance, when naively trained on "clean" data, is only slightly worse. We are strongly convinced the model will surpass the baseline by introducing errors to the training. Another fact to consider is that the acoustic data were collected using greedy decoding. In the final pipeline, a beam search with language model re-scoring will be used.

4. Spoken Language Translation

In the preceding chapter, we introduced and trained an "Enhanced ASR". The pipeline of the Enhanced ASR system consists of an "acoustic" model and a "translation" model. In the case of the ASR, the translation model rewrites phonetic transcriptions to graphemes in the language of the input.

In this chapter, we develop a spoken language translation pipeline for Czech and English language — in both directions. We build the pipeline architecture upon previously proposed Enhanced ASR. From the Enhanced ASR, we take the acoustic model, which remains identical. The essential difference is the change of target language for the translation model.

We organize this chapter as follows: first, we analyze associated work on SLT in Section 4.1. Next follows our motivation and experiment outline in Section 4.2. In Section 4.3 we describe training process and in Section 4.4 we evaluate the trained systems. Finally, we conclude the experiment in Section 4.5.

4.1 Related Work

With the advent of deep learning, we observe a clear trend towards end-to-end models in SLT.

For instance, Weiss et al. [2017] presents a attention-based model. They use the very architecture as ASR — encoder, and decoder. Their end-to-end trained network outperforms ASR-MT cascade on Spanish-English speech translation task.

Our great inspiration in our work and particularly for our spoken language translation, is the study of Salesky et al. [2019]. They examine an alternative speech feature representation for the end-to-end speech translation. The authors propose to use compressed phoneme-like speech representation. Their method first generates phoneme labels for a speech utterance. Second, consecutive frames with the same label are averaged-out. To compute the phoneme labels, authors use a hybrid HMM-DNN system implemented in Kaldi [Povey et al., 2011]. For the translation, they use a sequence-to-sequence model based on LSTM. The authors observe an improvement of the translation quality up to 5 BLEU on low- and high-resource language pairs.

4.2 Motivation and Experiment Outline

We propose to split the SLT pipeline to the acoustic and SLT model (the architecture is identical with Figure 3.1). As an intermediate representation, we choose phonemes. We are highly motivated by their success in the Salesky et al. [2019]. Also, as reviewed in earlier chapters on ASR (Chapters 2 and 3), phonemes are a competitive representation unit in speech processing.

In our spoken language translation pipeline, we take the acoustic model from the Chapter 2. Specifically, we use the ASR to the phonemes. As a translation model, we utilize the Transformer architecture.

We compare our proposed SLT system with a more conventional SLT pipeline — an ASR-MT tandem.

4.3 Training

Following the experiment outline as described in the preceding section, we train two translation models (Czech phonemes to English graphemes and English phonemes to Czech phonemes). Additionally, we also train baseline translation models (English and Czech, in both directions).

The training procedure remains almost identical to the training scheme in the Enhanced ASR (see Section 3.4.2). As we concluded in the Section 3.4.4, the bigger Transformer is a better fit for the ASR task. Furthermore, the translation task to a different language is even more difficult problem. Accordingly, we work only with the Transformer "big" in this chapter.

For evaluation during the training of phoneme-to-grapheme models we use `newstest2015`. We phonemize the source side of the data set using `phonemizer`.

Again, as determined in Section 3.3.5, we use separate vocabularies for source and target languages. Correspondingly, we use 32k vocabulary size. Source BPE is trained on a clean, phonemized CzEng dataset. Target is trained on clean original CzEng.

For training, we utilize following hyper-parameters for Transformer "big":

- GPU(s): 10 GPUs with at least 11 GB VRAM
- batch size: 2000
- learning rate: 0.03
- warm-up steps: 8000
- max steps: 600000 with manual abortion

To attain better regularization, we involve the BPE drop-out technique proposed by Provilkov et al. [2019]. We appoint the drop-out probability to 0.1, following the best setting from the original paper. The BPE drop-out is applied to both, the source and target of the translation.

The progression of training is in Figure 4.1 for Czech to English and in Figure 4.2 for English to Czech as evaluations on `newstest2015`.

4.4 Evaluation

In this section, we evaluate the trained models as part of the proposed spoken language translation pipeline (see Figure 3.1).

We evaluate an SLT pipeline consisting of an ASR and an NMT model. More precisely, our proposed pipeline includes an acoustic model (i.e. a model transcribing speech to phonemes) and a translation model (i.e., an NMT from phonemes in source language to graphemes in the target language).

Note, all models used in this comparison (baseline and proposed) are trained on the same datasets. The only contrast is the phonemization of transcripts for the proposed acoustic model and translation source for the proposed SLT translation model.

Again, as in the previous chapter, we apply different beam sizes during the evaluation.

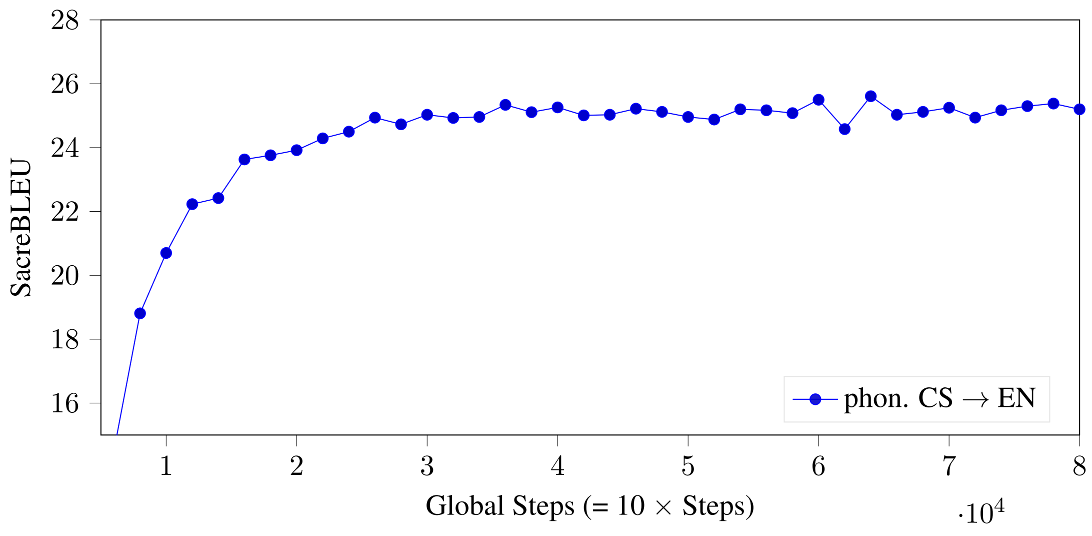


Figure 4.1: Evaluations on newstest2015 during the training. Phonemized Czech side as source and (original, in graphemes) English side as target.

4.4.1 SLT Test Set Preparation

To the best of our knowledge, there is no publicly available SLT test set from Czech to English. We therefore decided to create one.

We base our test set on a subset of the `newstest2018`. The original `newstest-2018` contains more than 3000 sentences. The test set is a collection of article passages from various news. There are articles of Czech origin, but also of English one. For our purposes, we selected only the articles of the Czech origin.

The re-speaker was instructed to use our online tool `Voice Recorder`.[1] The interface displays one sentence at time. The user can read the sentence, record it and replay. If she or he is not satisfied with the recording, the re-speaker may record a new version. With this workflow, the sentences are already segmented.

We employed one re-speaker — a Czech woman.

Because of limited resources, we were able to record 747 sentences. This resulted in 1:18:35 English, and 1:11:38 of Czech audio.

We evaluated the performance of our acoustic models on obtained test set. The results are in the Table 4.1.

We were quite surprised by the poor performance of Czech acoustic models — 34.33 % Phoneme Word Error Rate (PWER) and 34.51 % Word Error Rate (WER). We believe, this is due to the nature of the training data we used for the training of the models. The Corpus of Czech Parliament Plenary Hearings has a unique acoustic. Further, there is a high prevalence of man speakers. Ultimately, the size of the training corpus is rather small — only 400 hours.

On the other hand, the English models performed better — 18.61 % PWER and 18.60 % WER. First, the models are trained on approximately 2000 hours of speech. Also, the Common Voice dataset that is used for traing of the models contains many non-native speakers.

Still, the performance of the models is rather inferior to the test sets in well-knows datasets (such as LibriSpeech or Common Voice). We believe, this is partially due to the

[1]We developed this tool within the ELITR project (`elitr.eu`). Source code of the recorder is available at `https://github.com/ELITR/voice-recorder`

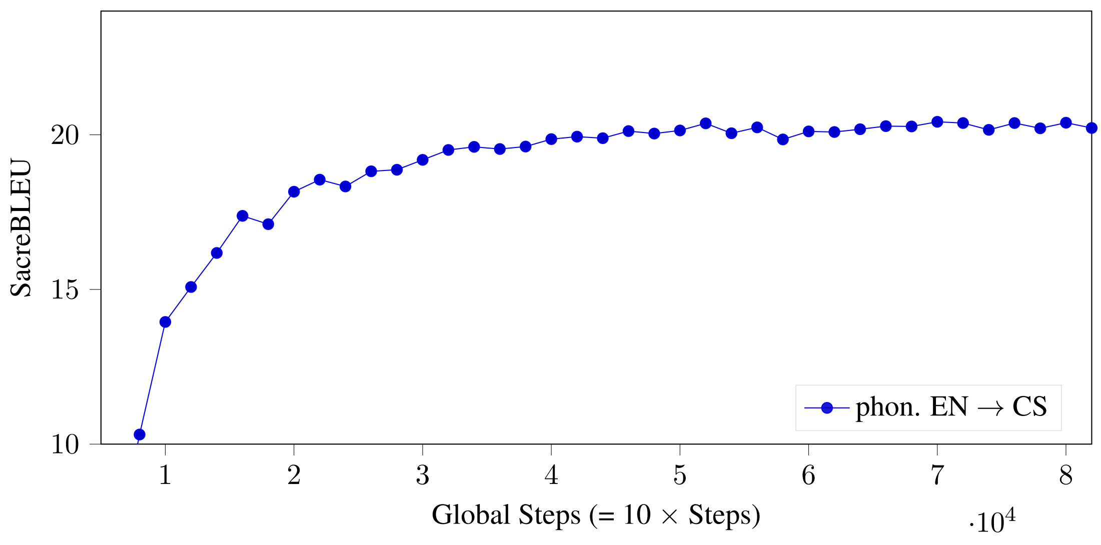


Figure 4.2: Evaluations on newstest2015 during the training. Phonemized English side as source and (original, in graphemes) Czech side as target.

	Model alphabet	Performance
Czech	phonemes	34.33 % PWER
	graphemes	34.51 % WER
English	phonemes	18.61 % PWER
	graphemes	18.60 % WER

Table 4.1: Evaluation of acoustic models on our test set derived from `newstest2018`.

high frequency of Czech proper nouns and specificity of the texts origin (news articles).

4.4.2 Beam Size

SacreBLEU scores in the Tables 4.5 and 4.9 are consistent with the established in the neural machine translation. We see that the models are, in general, confident at each step. Enlarging the beam size leads to performance deterioration. Having a beam size of one (effectively a greedy decoding) also leads to sub-optimal translations.

The only exception to this is the English to Czech proposed SLT. The beam size of 16 outperformed other beam sizes. However, the margin is relatively small — 0.23 BLEU. The model is probably less confident when the source is corrupted by the acoustic model.

4.4.3 Czech to English

For evaluation of the Czech to English SLT pipeline, we take the acoustic model from the Section 2.2. This model has Phoneme Word Error Rate 8.94 % on the phonemized Corpus of Czech Parliament Plenary Hearings.

We compare our proposal with a more typical setup — a standard ASR followed by an NMT. We employ the best performing ASR model from Section 2.1. This model has a Word Error Rate 7.64 % on the Corpus of Czech Parliament Plenary Hearings.

Table 4.5 contains SacreBLEU scores for Czech to English SLT. Translation samples are in Tables 4.3 and 4.4.

Additionally to the SLT pipeline evaluation, we also evaluate performance of the models on "translation" task. We take the true transcripts (phoneme transcripts for the proposed model and grapheme transcripts for the baseline model) and use them as the source of the SLT translation systems.

First, we evaluate the SLT task. We observe that the proposed phoneme-based system outperforms the baseline. The margin is rather small in terms of the SacreBLEU metric.

In the translation task (with correct transcripts as sources), the baseline closely outperforms the proposed system. This is probably due to the easier detection and consecutive translation of proper nouns in graphemes.

We see that the errors introduced by the acoustic/ASR model considerably derogate the translation quality — 11 points of SacreBLEU. A probable cause of such drastic SacreBLEU score reduction is high (phoneme) word error rate of the acoustic and ASR models (more than 30 %). This is quite surprising, as the re-speaker is Czech native.

Manual Evaluation We also manually assess the translation quality obtained in the SLT evaluation.

We design the evaluation procedure as a blind experiment. We use `QuickJudge`[2] and proceed as follows:

- we evaluate only a random sub-sample of the whole test set,
- we shuffle the order of segments (tuples consisting of the source, ground truth and hypotheses) in test set using `--shuffle` option,
- order of hypotheses in a segment is randomly switched (to prevent favoring one particular system across the test set),
- we evaluate the first 50 samples (note, the original order in the test set was previously destroyed),
- we annotate hypotheses as follows:
  - "$\star\star$" for correct translation,
  - "$\star$" for almost correct translation,
  - nothing for incorrect translation,
- we consider these criteria (in the given order):
  - content,
  - validity, or at least, phonological similarity of proper nouns,
  - grammar.

Distribution of manual annotations of the Czech to English SLT is in the Table 4.2.

As suggested by the SacreBLEU score, the proposed system outperforms the baseline in manual annotation. The proposed SLT system produces a correct translation in more cases than the baseline (12 correct, 14 almost correct vs. 7 correct and 15 almost correct). However, we must acknowledge that the overall performance of both systems

[2] `https://github.com/ufal/quickjudge`

	Correct	Almost correct	Incorrect
Proposed	12	14	24
Baseline	7	15	28
Only† proposed	7	4	7
Only† baseline	2	5	11
Both	1	6	17

Table 4.2: Distribution of ratings of manual annotation ofn the Czech to English SLT. We reviewed 50 randomly selected samples.
† Only means, that in case of (almost) correct translations the other system produced incorrect translation and vice versa.

Original	Tomáš Rosický	František Savov	Karim Rashid	Karel Štědrý
Proposed	Thomas Rozicki	Francis Sava	Cary Rashid	Karl the Generous
Baseline	Tomas Rizky	Frantise Saho	Cary Rash	Kanel the Generous

Table 4.3: Samples of names from Czech to English SLT.

is rather poor — in 24 cases (out of 50) the proposed and in the 28 cases the baseline failed to provide acceptable translations.

We discover a few notable differences while reviewing the translations. First, we find a better handling of proper nouns by the proposed system as very important. The translation are rarely without errors, but they are correctly recognized as proper nouns. Further, they tend to have a close pronunciation as the original.

Also, if the system fails to recognize a word, it is commonly made a proper noun (e.g., “Herec byl tehdy o 12 let starší...” translates as “Hannes was twelve years older...”). This is a bit distracting, but we see a possibility that in a broader context, the word can be understood by a user.

Interesting is also a more correct recognition on numbers. This is probably due to side-effect of the phonemization. The CzEng corpus is not consistent in using numbers. On the other hand, `phonemizer` rewrites all numbers written using numeric character to words.

Source	Obviněný podle všeho nad svým činem lítost neprojevoval.
Target	The accused apparently showed no regret for his actions.
Proposed	The accused seemed to show little remorse for his or her lack of pity.
Baseline	The accusation seemed to show no regret for his actions.

Source	Zakázku poprvé vysoutěžila v roce 2013, kdy ministerstvo průmyslu a obchodu vedl Martim Kuba ODS.
Target	The competition was first launched in 2013 when the Department of Industry and Trade was led by Martim Kuba of the ODS party.
Proposed	The contract was first won in 2013 when Martin Cuba OTS was in charge of the Ministry of Industry and Trade.
Baseline	She first competed for the contract in 2,13, when the Department of Industry and Trade led Martin Cuba from...

Source	Dvojice mladých Rusů Ilja Žirnov a Kira Čerkasovová byla pohřešována od prosince 2005.
Target	The two young Russians, Ilja Zhirnov and Kira Cherkasov, had been missing since December 2005.
Proposed	A pair of small Russians, Ilya Zirnov Akira Cherkasova, were missing since December 2005.
Baseline	Ilya Zernov, Akiracergasov, has been missing since December.

Table 4.4: Sentence samples from Czech to English SLT.

	Model	Source	Source error	Target	Cased, interpunct. Beam Size 1	4	16	Uncased, no interpunct. Beam Size 1	4	16
SLT (ASR source)	P2G	ASR Phon	PWER 34.33 %	En Graph	19.47	**20.07**	19.75	19.56	**19.96**	19.43
	baseline	ASR Graph	WER 34.51 %	En Graph	18.39	19.69	19.01	18.5	19.74	18.99
Translation (clean source)	P2G	Clean Phon	0 %	En Graph	30.14	30.82	30.35	30.44	30.82	30.56
	baseline	Clean† Graph	0 %	En Graph	30.31	**31.07**	30.45	30.74	**31.55**	30.77

Table 4.5: Evaluation of the proposed Czech to English model (phonemes to graphemes — P2G) and the Czech to English baseline (graphemes to graphemes). We evaluate performance on SLT and Translation task. SLT task obtained source from ASR transcripts. Translation task is done on clean (original) source.
† ASR-like Graph is original lowercase source with stripped interpunction.

	Correct	Almost correct	Incorrect
Proposed	16	21	13
Baseline	18	18	14
Only† proposed	6	2	7
Only† baseline	6	1	8
Both	4	11	6

Table 4.6: Distribution of manual annotation of the English to Czech SLT. We reviewed 50 randomly selected samples.
† Only means, that in case of (almost) correct translations the other system produced incorrect translation and vice versa.

4.4.4 English to Czech

In order to evaluate the English to Czech translation model as part of the proposed SLT pipeline, we take the English acoustic model from Section 2.2. The model has a phoneme word error rate 4.18 % on LibriSpeech `test-clean` and 11.48 % on the `test-other`.

The baseline SLT pipeline uses as ASR model the Jasper model trained on LibriSpeech and Common Voice. We downloaded the model from the NVIDIA NGC.[3]

We also assess the translation models performance on the "translation" task (i.e., the translation sources are true transcripts, not burdened with errors introduced by the acoustic/ASR models).

Table 4.9 contains SacreBLEU scores of the English to Czech translation models as part of a SLT pipeline and in the translation task. Translation samples are in Tables 4.7 and 4.8.

Similarly, as the Czech to English SLT, the proposed system (with phonemes as an intermediate step) outperforms the baseline. Again, the difference is rather small — 0.52 BLEU.

We manually review the quality of the both SLT systems. The distribution of manual annotations of the Czech to English SLT is in the Table 4.6.

Compared to the opposite direction (Czech to English), the overall subjective translation quality improved significantly. We found only 13 and 14 translations respectively incorrect. Although the direction to Czech language is more challenging in general XXX do I need to support this?, we explain this reduction of incorrect translations as the result of a better acoustic and ASR model.

In the random sub-sample that we evaluated, we found the translations of the baseline to be slightly better. Still, the difference is relatively small. The baseline produced more "correct" translations, while the proposed produced less "incorrect" ("correct" + "almost correct") translations.

We discovered that the baseline tends to mark unknown words as proper nous (e.g.: "This species lives..." translates as "Tento Pecies žije...", or "...the murderer's parents..." as "rodičů Murdereora"). We marked such translation as "almost correct", as we think that such words can be understood in the context. Similarly, the proposed system does so, but more frequently around proper nous (e.g., "...in the new Prima series Kapitán Exner" as "...v nové Primma Series Capitan Expert").

[3]`https://ngc.nvidia.com/catalog/models/nvidia:multidata%20set_jasper10x5dr`

Original	Mádlová	Donald Trump	Lucie Bílá	Simona Krainová
Proposed	Madlova	Donald Tramp	Rusi-Billa	Simana Cranova
Baseline	Madlovová	Donald Trumf	Rutiabilia	Simona Cranova

Table 4.7: Samples of names from English to Czech SLT.

Source	Now it is a finished product that is produced in our country.
Target	Nyní jde o hotový produkt, který je navíc vyráběný u nás.
Proposed	Nyní je to hotový produkt, který se vyrábí v naší zemi.
Baseline	Nyní se jedná o hotový produkt vyráběný v naší zemi.
Source	An extensive fire damaged the Torch skyscraper in Dubai in February 2015, ruining more than 100 apartments.
Target	Rozsáhlý požár poškodil dubajský mrakodrap The Torch rovněž v únoru 2015, zničil přes sto bytů.
Proposed	Rozsáhlý požár poškodil mrakodrap v Dubaji v únoru 2015 a zničil více než sto bytů.
Baseline	Rozsáhlý požár poškodil průmyslovou výrobu mrakolestných mrakodrapů v únoru, dva tisíce patnáct a zničil více než sto bytů.
Source	Dehydration manifests itself as a complete loss of self-sufficiency.
Target	Dehydratace se projeví jako naprostá ztráta soběstačnosti.
Proposed	Hydratace se projevuje jako naprostá ztráta sebeúčinku.
Baseline	Dehydratace se projevuje jako úplná ztráta soběstačnosti.

Table 4.8: Sentence samples from English to Czech SLT.

With the overall translation quality, the handling adequacy of proper nouns also rises (see the Table 4.7). It is worth noting that the re-speaker was a native Czech. The Czech proper nouns are hence correctly pronounced in Czech. Therefore, we find it quite interesting, that both system were able to translate the name satisfactorily.

	Model	Souce	Souce error	Target	**Cased, interpuct.** Beam Size 1	**Cased, interpuct.** Beam Size 4	**Cased, interpuct.** Beam Size 16	**Uncased, no interpunct.** Beam Size 1	**Uncased, no interpunct.** Beam Size 4	**Uncased, no interpunct.** Beam Size 16
SLT (ASR source)	P2G	ASR Phon	PWER 18.61 %	Cs Graph	16.64	17.12	**17.35**	14.28	14.76	**14.9**
	baseline	ASR Graph	WER 18.80 %	Cs Graph	16.51	16.83	16.68	14.29	14.53	14.42
Translation (clean source)	P2G	Clean Phon	0 %	Cs Graph	21.75	22.32	22.26	19.6	20.01	20.06
	baseline	Clean† Graph	0 %	Cs Graph	21.85	**22.47**	22.21	19.74	**20.09**	20.07

Table 4.9: Evaluation of the proposed English to Czech model (phonemes to graphemes — P2G) and the English to Czech baseline (graphemes to graphemes). We evaluate performance on SLT and Translation task. SLT task obtained source from ASR transcripts. Translation task is done on clean (original) source.
† ASR-like Graph is original lowercase source with stripped interpunction.

4.5 Conclusion

We proposed an SLT pipeline with phonemes as intermediate representation. We successfully trained the models for English and Czech (in both directions). Further, we compared these models with baselines (traditional ASR to graphemes followed by an NMT model). In both direction, the proposed SLT pipeline achieves slightly higher SacreBLEU score.

We also assessed the translation quality subjectively using a random experiment. In this case, the baseline system performed better in the direction from English to Czech, and in the Czech to English direction performed the proposed SLT system better.

We believe that the proposed system has a potential in the future. Therefore, we would like to review the adaptation of the system to the errors produced by the acoustic model. Also, other techniques known from neural machine translation can be applied (e.g., backtranslation). We are convinced, the proposed systems are capable of even better translation quality.

5. Fine-tuning Enhanced ASR

In Chapter 3, we proposed the "Enhanced ASR" pipeline (see Figure 3.1) which consists of an *acoustic* model and a *phoneme-to-grapheme translation* model. So far, we used "clean" (without any errors) training data to train the translation model. The obvious problem is that the phoneme transcripts produced by the acoustic model in the ASR pipeline will most probably contain errors, to which the translation model is not ready.

In this chapter, we review the fine-tuning of the ASR translation model to errors introduced by the acoustic model.

This chapter is organized as follows: in Section 5.1 we review the related work. Sections 5.2 and 5.3 describe process in which we obtain training data with "ASR errors". Finally, Section 5.4 reviews setups for final part of proposed Enhanced ASR pipeline and trains it.

5.1 Related Work

Hrinchuk et al. [2019] deal with the correction of errors in ASR by introducing a Transformer post-processing step.

The authors first train an ensemble of 10 Jasper ASR models on the LibriSpeech dataset. Using these models, they collect "ASR corrupted" data (pairs consisting of a transcript from an ASR ensemble model and a correct transcript). To collect more training data, they additionally allow drop-out in the Jasper models during the inference. They also use a Cutout [DeVries and Taylor, 2017] technique that randomly cut rectangles out of the spectrogram. The authors keep pairs with unique ASR transcripts and remove pairs with a word error rate higher than 50 %.

Subsequently, they train a Transformer model on this data. The ASR transcripts serve as the source and the original true transcripts as the target. In their best setup, they utilize transfer learning, where they take BERT [Devlin et al., 2018] (notably, a masked language model consisting only of a Transformer encoder) and initialize both encoder and decoder of their Transformer correction model with the BERT's weights.

5.2 English "Corrupted" Training Data

In this section, we describe the process in which we gather "corrupted" data from the acoustic model. We will use these data later to improve the robustness of the phoneme-to-grapheme translation model.

We design the setup similarly, as described in Section 5.1. First, we train ten acoustic models. To obtain more data, we additionally store checkpoints during training and keep the last four of them (yielding 40 models). Because of the resource scarcity, we decided to use the QuartzNet architecture instead of Jasper.

Second, we transcribe all available training data using the set of models obtained in the previous step. Subsequently, we pair the "corrupted" transcripts with the correct transcripts. Similarly to the Hrinchuk et al. [2019], we apply drop-out and the Cutout during the inference. We obtain a parallel corpus of "corrupted" and "clean" sentences. We keep only tuples with unique ASR transcripts and remove pairs with PWER higher than 50 %.

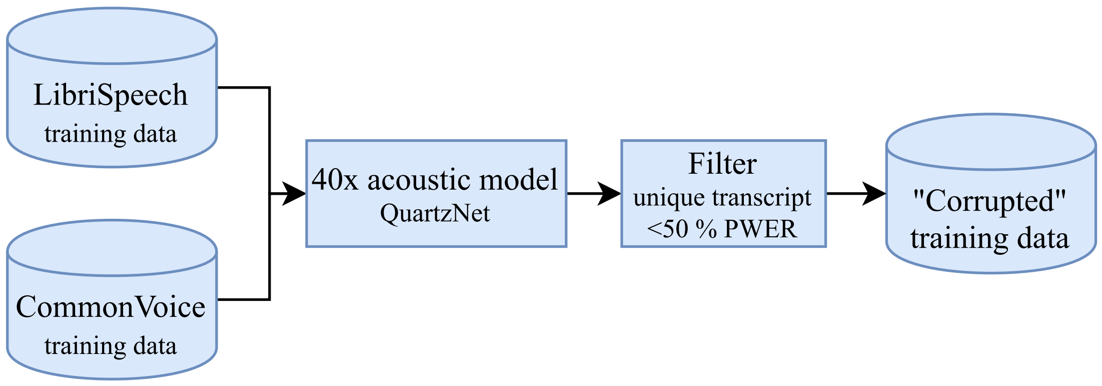


Figure 5.1: Overview of setup for obtaining “corrupted” training data.

These data will then be used in further training as a source of “natural” noise that occurs in speech recognition.

Overview of the setup is pictured in Figure 5.1.

5.2.1 Data Preparation

To train the ensemble of the acoustic models, we use LibriSpeech and Common Voice datasets. We concatenate these two and divide them into ten folds of the same size. We deliberately not shuffle the concatenated dataset prior to splitting it into folds, so that the difference among the trained models is as much as possible (proportion of training data from LibriSpeech and Common Voice will vary more). The models are trained on these folds in a cross-validation manner: i-th model skips i-th fold during training.

5.2.2 Ensemble Training

We train ten models. Additionally, we store checkpoints every 5000 steps. Instead of bigger Jasper, we choose QuartzNet. Nevertheless, they are similar enough, and we assume they behave likewise. The main reason for our choice are reduced hardware requirements and hence faster convergence. In contrast with the bigger ASR Jasper model that we train on 10 GPUs, each QuertzNet model for data collection is trained only on one GPU. After less than a day of training, the models perform almost as good as the bigger model.

We utilize transfer learning to reduce the training time. For English, the QuartzNet encoders are initialized with checkpoint available at NVIDIA NGC.[1] The model was trained on LibriSpeech and Common Voice. As the target vocabulary differs for our setup (phonemes instead of graphemes), we apply the method from chapter 2. The training is divided into the two phases: (1) Decoder adaptation phase: the encoder is initialized with the pre-trained weights and is frozen; the decoder is randomly initialized. Only the decoder is trained. (2) Full training: the encoder is unfrozen and trained together with the decoder. The adaptation phase takes 2000 steps, and full training continues afterward for another 30000 steps.

[1]`https://ngc.nvidia.com/catalog/models/nvidia:quartznet15x5`

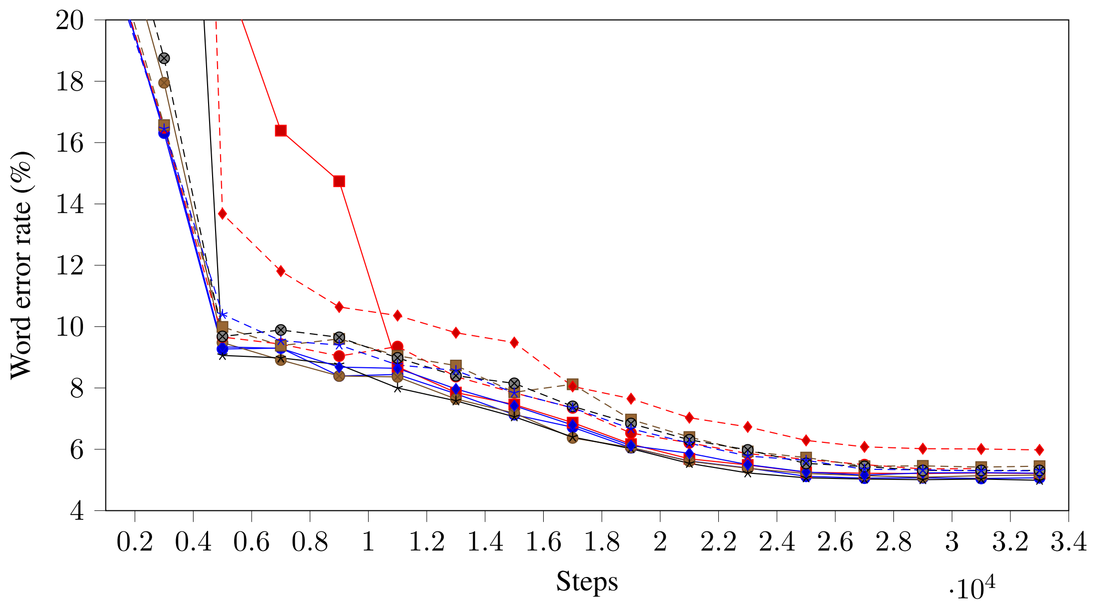


Figure 5.2: Evaluation on LibriSpeech `dev-clean` during the training of ten models (using greedy decoding). One epoch takes approximately 24400 steps.

Model	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Adapt. phase	15.17	15.02	16.66	22.15	15.07	15.39	15.24	17.44	15.38	33.05
Full training	5.07	5.21	5.16	4.99	5.22	5.34	5.45	5.31	5.30	5.98

Table 5.1: Results in % of word error rate (using greedy decoding) on LibriSpeech `dev-clean` for all trained models.

5.2.3 Acoustic Models Performance

On average, after the first adaptation phase, the word error rate of the most models on LibriSpeech `dev-clean` dropped under 16 % (this took approximately 5 hours). One model had WER two times worse than others (33.05 %), and one did not converge at all. After the full training, which took about 15 hours, the average WER is 5.3 % with very small variance. Compared with big Jasper, it is only about 1.5 % points more. Evaluations during the training can be seen in Figure 5.2 and the final results are shown in Table 5.1.

5.2.4 English Corrupted Data Collection

In order to generate more data, we make checkpoints during training every 5000 steps and keep the last four for every model. This gives us 40 unique models. Additionally, we employ drop-out and Cutout to gain more variability in the collected data.

Concatenated LibriSpeech and Common Voice training data are used as the source for the inference. Using all 40 models, we yield 36M sentence pairs. We keep only the pairs with unique source sentence (i.e., has a unique error) and destroy pairs with PWER over 50 %. From the 36 million tuples, we get 7M filtered sentence pairs, particularly 3.7M from LibriSpeech and 3.3M from Common Voice.

Distribution of PWER in training and development sets is in histogram 5.3. We filter out all pairs with PWER higher than 50 %. As we can see in the histogram 5.3, only 7 % of all collected training data is left out. We can observe that the distribution of PWER

Set	AVG WER	Median WER	STD
Training	10.24 %	2.70 %	16.64 %
LibriSpeech dev-clean	4.33 %	0.0 %	8.37 %
Common Voice dev	11.98 %	0.0 %	17.71 %

Table 5.2: Performance of the 40 ensemble models on the datasets.

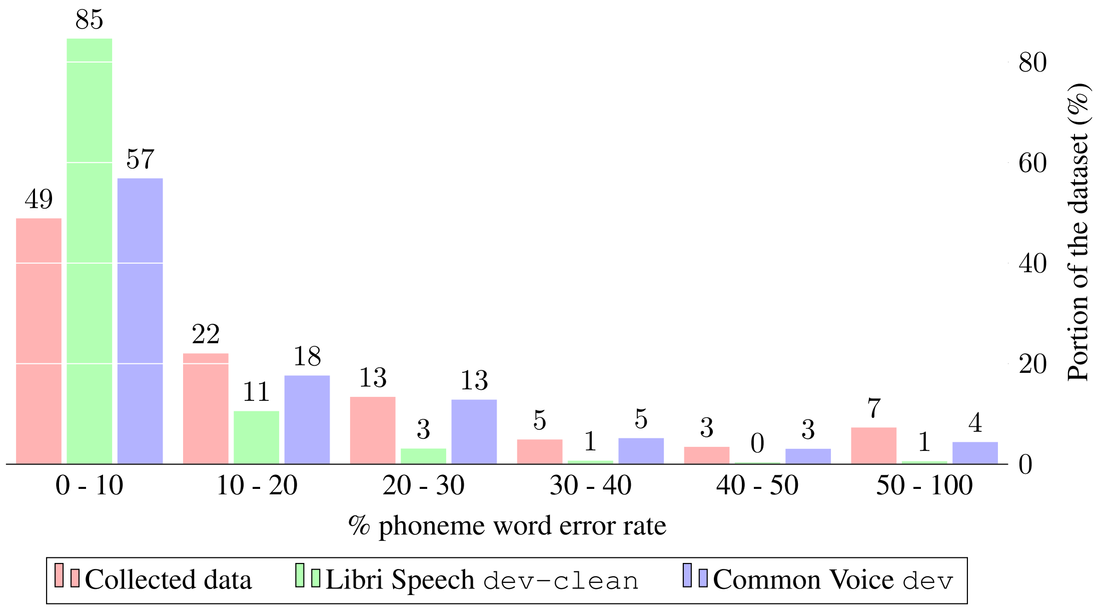


Figure 5.3: Distribution of PWER on the collected English "corrupted" data and the two development sets.

for training data almost copy the distribution of Common Voice dev with training data having slightly more pairs with smaller PWER. On the other hand, LibriSpeech dev-clean has significantly more examples with small PWER.

5.3 Czech "Corrupted" Training Data

In this section, we reproduce the previously described experiment for the Czech language. Most challenging is to overcome the scarcity of speech data. The Czech speech corpus has approximately 400h. On the other hand, the two English corpora yield together almost 2000h.

5.3.1 Task Setup

We proceed similarly to the setup described in Section 5.2: we split the available data into "folds". Each fold then serves as a training corpus for one acoustic model. Unlike for the English, we decided to have only five folds instead of 10 to take the dataset size into account.

The training and collection of corrupted data remain the same.

5.3.2 Ensemble Training

Following the receipt from English, we employ QuartzNet architecture, train all models on one GPU, and follow the same transfer learning technique. We train-off from our best performing, fully trained, Czech ASR model (see Chapter 2). Because of the Czech training data scarcity, we train only five folds. For more details regarding the training, see Section 5.2.

Figure 5.4 depicts the performance of the trained acoustic models on the development set during the training.

5.3.3 Acoustic Models Performance

Table 5.3 offers detailed training results for all models.

We observe a dramatic PWER decline at the beginning (until step 12k), followed by almost no change until the end of the training. We assume there are two reasons for this behavior: (1) data scarcity; (2) contrast in grapheme-to-phoneme correspondence in the Czech and the English language. We assume the latter to have a more significant impact, as the model converged after 7 thousand steps, which is roughly right after seeing all examples once (2000 steps adaptation phase + one epoch of 4775 steps of the full training).

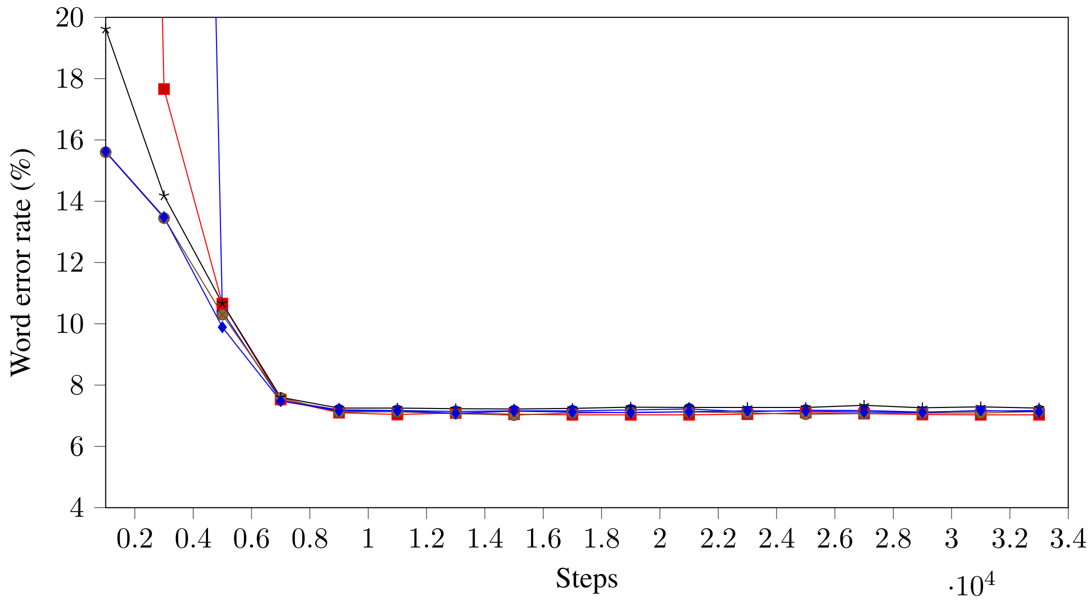


Figure 5.4: Evaluation of the development set during the training of the 5 models (using greedy decoding). One epoch is approximately 4750 steps.

Model	**1**	**2**	**3**	4	**5**
Adapt. phase	97.91	17.63	13.53	14.09	13.37
Full training	7.17	7.03	7.14	7.25	7.13

Table 5.3: Results in % of PWER (greedy decoding) on the development set for all trained models.

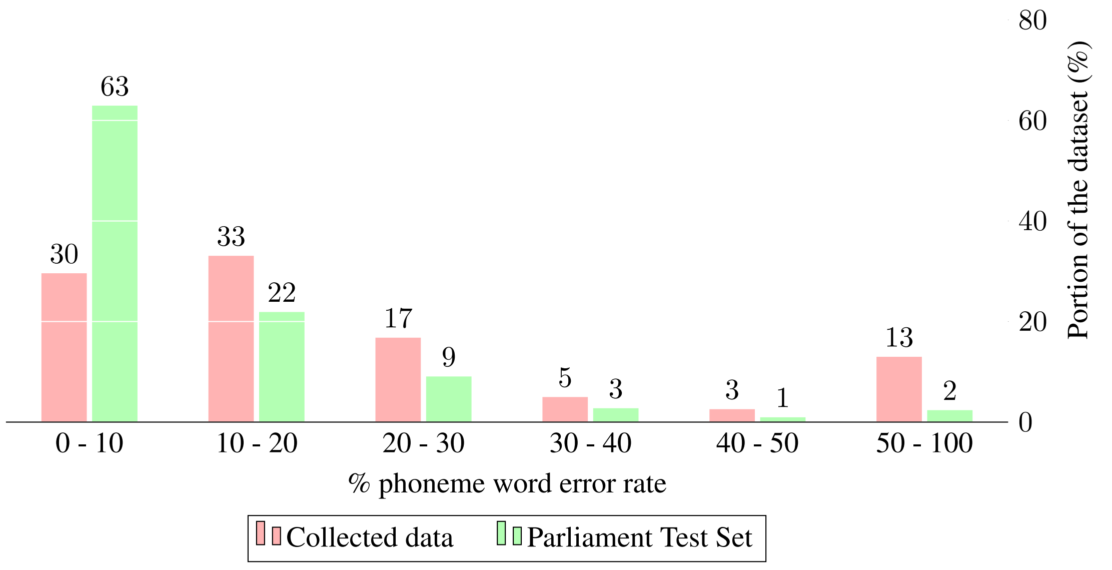


Figure 5.5: Distribution of PWER of the collected Czech "corrupted" data and the Czech Parliament Hearings test set. Note, we used the test set instead of the development set. PWER of the recordings in the development set does not correspond to the actual sentence-level distribution of PWER. This is because the development set contains 12 further unsegmented recordings each more than 10 minutes long.

5.3.4 Czech Corrupted Data Collection

In order to generate more data, we make checkpoints during training every 5000 steps and keep the last four for every model. This gives us 20 unique models. Additionally, we employ drop-out and Cutout to gain more variability in the collected data.

We use the training data of the Czech Parliament Hearings dataset as the source for the inference. Using all 20 models, we yield 3.8M sentence pairs. This is almost ten times less than for English. We keep only the pairs with unique source sentences (i.e., has a unique error) and destroy pairs with PWER over 50 %. Finally, we get two million sentence pairs.

Distribution of PWER in training and dev sets is in histogram 5.3. We filter out all pairs with PWER greater than 50 %. As we can see in the histogram 5.3, only 13 % of all collected training data is removed. We can observe that the collected data contains more recordings with higher PWER than the test set.

5.4 Fine-Tuning of Enhanced ASR

In this section, we use the collected "corrupted" data to enhance the phoneme-to-grapheme translation models. We experiment with training and fine-tuning using this data. We also examine various transfer learning schemes.

5.4.1 Experiment Outline

We design the fine-tuning experiment similarly to that of Hrinchuk et al. [2019] described in Section 5.1. However, we must consider the differences in our ASR pipeline.

Our ASR pipeline consists of an acoustic model transcribing the speech into phonemes, and a phoneme-to-grapheme translation model. On the other hand, the authors have an ASR model to graphemes and a correction model.

In our experiments, we initialize the model weights from SLT tasks. We find the SLT task as a promising starting point for enhancing the ASR transcripts. We base our hopes on the fact that the main task of the encoder in the SLT is to create an abstract representation of the input sentence based on its meaning. This means the encoder must consider the content and semantics of the sentence. Similarly, the decoder must decode the content given by the encoder and is also responsible for creating a meaningful sentence in the target language. This is in contrast with what the Transformer model does when trained as a phoneme-to-grapheme translation model operating within one language. Based on our experience from Chapter 3, we suspect the model remembers rules for rewriting the phonemes to graphemes without considering the context.

For the English ASR, we experiment with three different setups:

Correction Baseline As a baseline, we train the translation model from scratch. Considering the over-fitting that we observed in Section 3.3 (i.e., when the translation model is trained from scratch solely on corrupted data), we decided to mix (one to one) the corrupted training data with clean phonemized CzEng data.

SLT Transfer In this setup, we proceed exactly as in the *Correction Baseline*. Additionally, we initialize the Transformer encoder from English to the Czech SLT translation model, and the Transformer decoder is initialized from the Czech to English SLT translation model (see Chapter 4 for more details on SLT).

SLT + BERT Transfer This setup utilizes a checkpoint from SLT for the encoder and the pretrained BERT for the initialization of the decoder. We do not initialize the encoder as the source of the model are phonemes (thus, we would have to change the tokenizer). We hypothesize this would be a significant change of the task for the pre-trained BERT, and it would diminish the advantage of the pre-training.

There are several variants of BERT. We decided for `large-uncased`. We chose the `large` to match the Transformer encoder (that we initialized from SLT) model dimension (1024).

Additionally, we use only corrupted training data from the LibriSpeech dataset for the fine-tuning.

For the Czech ASR, we experiment with two setups:

Correction Baseline We train the Czech ASR phoneme-to-grapheme translation model baseline as the English counterpart. We train the model from scratch using a one to one mix of clean phonemized CzEng and corrupted training data (to overcome the over-fitting).

SLT Transfer We initialize the Transformer model with weights from the SLT task. The encoder is initialized with the encoder from the Czech to English SLT. The decoder is initialized with the decoder from the English to Czech SLT.

5.4.2 Training/Fine-Tuning

We train the models on two GPUs with 16 GB VRAM. We set the batch size to 8000 tokens, learning rate to 2×10^{-3}, and 16000 warm-up steps. The length of the training is set to 6000000, but we manually abort it after the convergence.

All phoneme-to-grapheme models share the same architecture (the Transformer `big`) except for the *SLT + BERT Transfer*. In this case, the decoder copies the BERT `large-uncased` architecture (24-layer, 1024-hidden, 16-heads).

For the *SLT + BERT Transfer*, we tried the same training procedure as for other setups. However, we were unable to obtain any reasonable performance (we got WER of 28% on LibriSpeech `dev-other`). We hypothesize this is due to the vast amount of weights that must be randomly initialized in the decoder. The BERT model is essentially a Transformer encoder. Hence it does not have the encoder-decoder attention layer, which must be randomly initialized. During the training of the whole model with many randomly initialized weights, the initially trained weights from the BERT might depart too far from the optimum.

To overcome this issue, we use an analogous adaptation trick as for the training of the acoustic model (see Section 2.2). We freeze all weights initialized from seed models and train only the randomly initialized weights until the convergence (the criterion was the loss on the validation dataset). This adaptation takes 13500 steps in our case. Subsequently, the training of the whole model continues as in the experiments.

Oversight on the training provides the LibriSpeech `dev-other` for English, and the Parliament Hearings `test` for Czech. The overview is in Figures 5.6 and 5.7. We choose the test set instead of the development set for the Czech as the development set consists of unsegmented long recordings that do not fit into the Transformer model.

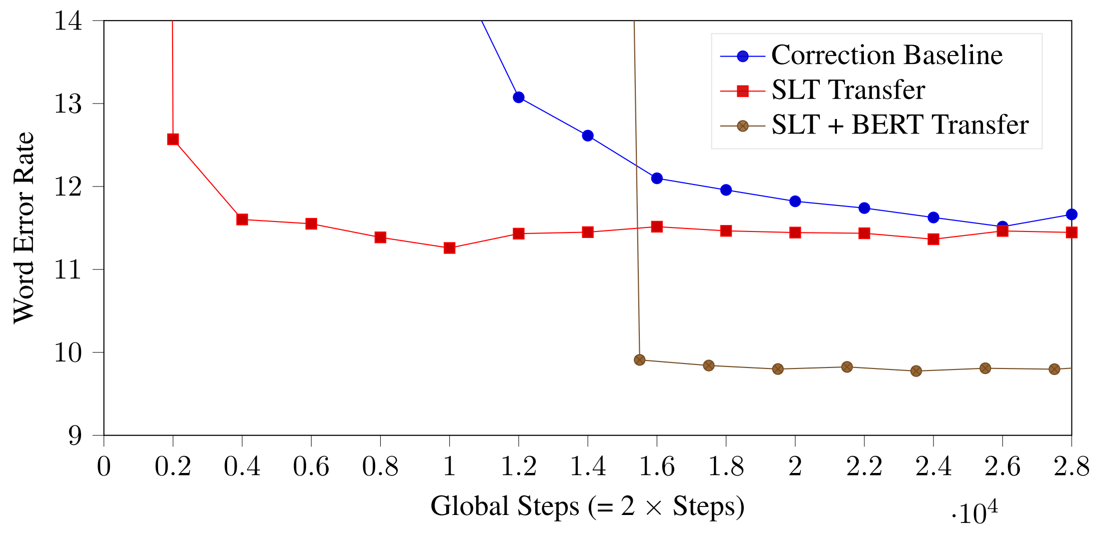


Figure 5.6: Training/fine-tuning of the English phoneme-to-grapheme ASR models. Performance on the LibriSpeech `dev-other`.

5.4.3 Performance Evaluation

In this section, we evaluate the performance of the models. For English, we utilize Common Voice `test` set and LibriSpeech `test-clean` and `test-other`. For

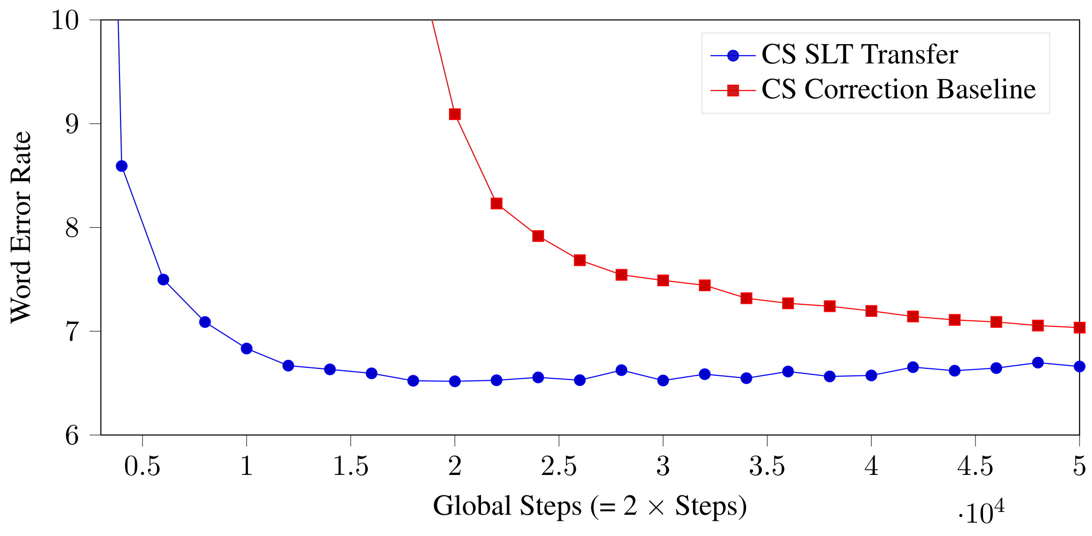


Figure 5.7: Training/fine-tuning of the Czech phoneme-to-grapheme ASR models. Performance on Parliament Hearings `test`.

Czech, we use the test set from Parliament Hearings corpus. The results are in Tables 5.4 and 5.5.

English From the results in Table 5.4, it is evident that the training/fine-tuning on corrupted data helps.

Particularly, the *Correction Baseline* and *SLT Transfer* are both trained on data that contains erroneous data from the Common Voice dataset. Both models perform on the Common Voice `test` very well (3.91 and 3.26 % WER respectively). On the other hand, the *SLT + BERT Transfer* performs on the Common Voice substantially worse (even worse than the *Clean Baseline*).

For the LibriSpeech test sets however, we see that the models perform worse when trained on corrupted data containing Common Voice transcripts. The *SLT + BERT Transfer*, which is fine-tuned only on corrupted data from LibriSpeech, performs better than the baseline.

On our `read-newstest` (for details on the test set see Section 4.4.1), the *Correction Baseline* and *SLT Transfer* perform better than the *Clean Baseline*. It seems that both models learned to correct some of the errors. Differently, the *SLT + BERT Transfer*, which was fine-tuned on LibriSpeech corrupted data, performs worse than the *Clean Baseline*. We assume that this is due to the slightly over-fitting of the model to the LibriSpeech corpus.

In the case of *SLT Transfer*, we observe a clear advantage of the initialization from the SLT task over the random initialization (*Correction Baseline*). The training converges rapidly faster, and the final performance is better compared with the random initialization.

Czech The results in Table 5.5 suggest the same as for the English.

Training using corrupted data enhance the ability of the model to recover from errors introduced by the acoustic model (both models surpass the *Clean Baseline*).

Furthermore, it is clear that initializing the models from the SLT task helps in two ways. First, it significantly reduces the training time (by more than 20k global steps —

	Common Voice `test`	LibriSpeech `test` `clean`	LibriSpeech `test` `other`	`read-newstest`
Clean Baseline	9.72	4.87	11.67	16.38
Correction Baseline	3.91	5.25	11.81	15.56
SLT Transfer	**3.26**	5.10	11.75	**15.46**
SLT + BERT Transfer	12.93	**4.13**	**10.21**	19.00

Table 5.4: Performance of the English ASR phoneme-to-grapheme translation models.

	Czech Parliament Hearings `test`	`read-newstest`
Clean Baseline	9.60	33.22
Correction Baseline	7.02	31.94
SLT Transfer	**6.76**	**29.74**

Table 5.5: Performance of the Czech ASR phoneme-to-grapheme translation models on the Czech Parliament Hearings corpus test set.

see Figure 5.7). Second, it alleviates the final performance of the model compared to the *Clean Baseline* but also *Correction Baseline*.

5.5 Conclusion and Future Work

In this chapter, we reviewed a performance-enhancing technique for the proposed ASR pipeline.

First, we gathered “corrupted” acoustic data. We trained an ensemble of acoustic models for each language (English and Czech). Consequently, we used these models to gather the corrupted data from speech corpora.

Second, we utilized this data for the training/fine-tuning of the phoneme-to-grapheme translation models from the proposed ASR pipeline. We experimented with different setups. For example, we considered training on mixed “clean” and “corrupted” data or different transfer learning schemes.

We observed that training-off the checkpoints from the spoken language translation task is the best option. Based on this observation, we suspect that a multi-task training scheme combining SLT and ASR tasks for a language tuple could be promising (i.e., a shared English encoder for English ASR and English-to-Czech SLT decoders, and vice versa).

Further, it is clear that training using “corrupted” data helps to reduce the WER. However, the method for obtaining such data is restricted to speech corpora (speech and transcriptions). This is too constraining as speech corpora are generally less available than, for example, free texts (recall, for the training of ASR phoneme-to-grapheme translation we use one side of the CzEng — phonemized as the source and original as target) which are effortlessly available in almost unlimited amounts. Besides, this scheme is even harder to apply for the SLT, as the speech-to-translation corpora are rare. Finally, the utilization of acoustic models is “too expensive” on hardware resources. To overcome these limitations, we see a potential for researching a method that would be able to simulate the errors produced by the acoustic model.

6. Online Adaptation

In this chapter, we experiment with online adaptation of the proposed enhanced ASR pipeline.

We assume that many ASR errors are systematic and highly dependent on the particular speaker, for example, due to his/her mother tongue. We expect that some of these errors can be identified and corrected in the ASR phoneme-to-grapheme translation model. We plan to identify these corrections and extract simple speaker-specific rules.

This chapter is organized as follows: in Section 6.1 we propose online adaptation and incorporate it into the proposed ASR/SLT pipeline. In Section 6.2, we experiment with the online adaptation. Finally, in Section 6.3 we conclude the chapter.

6.1 Enhanced ASR/SLT Pipeline with Online Adaptation

We propose the enhanced ASR pipeline with online adaptation as follows: the acoustic model outputs the phoneme transcripts. The phoneme transcripts are adapted using the Online Adaptation model. The adapted output (still in phonemes) is fed into the phoneme-to-grapheme translation model (note, under the term translation we mean both the monolingual — phonemes to graphemes in the same language — and translation from source language phonemes to target language graphemes). The translation model translates the phoneme input into graphemes using beam search. The model outputs the best translation candidate from the beam as a translation of the whole ASR pipeline. All beam translations are passed through the `phonemizer`, and the online adaptation model learns new rules from the transcripts.

In the case of the SLT pipeline, two parallel phoneme-to-grapheme translation models are needed: the ASR and the SLT. The ASR translation model would be used for obtaining "training data" for the online adaptation model.

The schema of the proposed enhanced ASR pipeline with online adaptation is in Figure 6.1.

6.1.1 Online Adaptation Model

We seek a method that would be capable of quick, on-the-fly adaptation as a speaker talks. The apparent requirement of such online adaptation model is that it must be able to learn quickly. Hence, we rule out neural networks as they require a higher volume of training data. Short of such data, the neural networks tend to over-fit the examples.

Therefore, we decided on the rule-based model. We take our inspiration in the work we previously reviewed for phoneme-to-grapheme models (see Section 3.1.1). More precisely, the work of Horndasch et al. [2006]. Their main objective is to find appropriate orthographic representations for phoneme strings. Our use case differs as we seek phoneme-to-phoneme mapping correcting errors instead of phoneme-to-grapheme mapping. Using an expectation-maximization algorithm [Dempster et al., 1977], we seek the best "correct-to-incorrect" phoneme alignment (in their setup, they are looking for best phoneme-to-grapheme alignment). In the second step, we cluster neighboring symbols together to account for the insertions (i.e., to remove "ϵ to phoneme" rules).

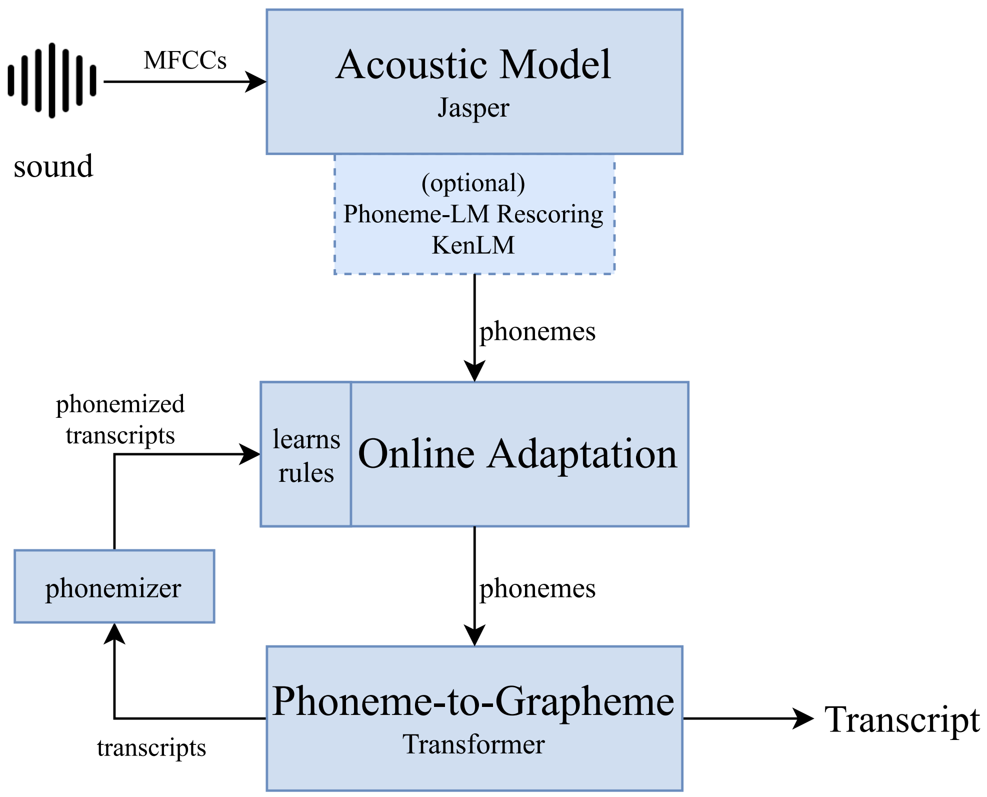


Figure 6.1: Enhanced ASR pipeline with online adaptation.

Finally, n-gram probabilities of symbol pairs are learned. During the inference, the input string is split into individual phoneme symbols. For each symbol are generated all possible symbol pairs. According to the beam width, the best sequences are taken.

Model Training As already described, we learn the phoneme rewrite rules from aligned "incorrect-to-correct" phoneme transcriptions. To align these transcriptions, we utilize the expectation-maximization algorithm as follows:

1. Compute the initial `alignment` using the Levenstein algorithm (insertion, deletion, and substitution operations are considered, each with the same cost).
2. *While* substitution frequencies change do:
   (a) **Expectation step:** based on the `alignment`, compute the substitution frequencies,
   (b) **Maximization step:** align training pairs based on frequencies using dynamic programming.

The last computed alignment defines the rewrite rules: "rewrite the symbol from the source string (the "incorrect" one) to the symbol on the corresponding position in the target string (the "correct" one)". The insertions in the alignment are represented with special ϵ character in the source string (insertion, ɪnsɜːʃən):

ɪnsɜː$\epsilon\epsilon\epsilon$
ɪnsɜːʃən

We do not allow rules "$\epsilon \rightarrow \ldots$", because they would allow infinite string growth during the beam search decoding. Instead, we cluster the rules with ϵ symbol as head with neighbors using following algorithm:

For each `rule` in `alignment`:

1. If `rule` is *empty*: join with the previous or next `rule`
2. Else: add to `new_rules`.

From the above insertion example, we would extract following rewrite rules:

$$\text{ɪ} \rightarrow \text{ɪ} \mid \text{n} \rightarrow \text{n} \mid \text{s} \rightarrow \text{s} \mid \text{ɜː} \rightarrow \text{ɜːʃən}.$$

Finally, we train an n-gram model. The purpose of the n-gram model is to learn interactions between the neighboring phoneme rewriting rules. We utilize the KenLM for this task. As the KenLM is word-oriented, we dump the rules as separate words in form "source-target". We substitute the space symbol with an underscore.

Inference We use the trained n-gram model during the inference — rewriting input using the rules. For this, we utilize beam search:

For each symbol from the input:

1. generate candidates using rules with corresponding source symbol (by appending rule tail to all beams),
2. score the candidates using the n-gram model,
3. keep top w candidates according to the beam width w.

6.2 Experiments

In this section, we experiment with different variations of the proposed online adaptation algorithm.

For testing of the adaptation model, we use the parallel Czech - English ASR/SLT test set `read-newstest` (see Section 4.4.1 on page 50). To develop the adaptation model independently on the phoneme-to-grapheme model, we use instead of corrected transcripts that would be given by the P2G model the golden transcripts. Additionally, we create "fake" acoustic output with artificial errors to better analyze the behavior of the model.

We simulate "online" adaptation by iterating over the test set with the step of 100 sentences, i.e., in each iteration, the training set for the adaptation model grows by 100 sentences, and the model corrects the next 100 sentences.

6.2.1 Naïve Approach: Train & Rewrite All

First, we experiment with a rather "naïve" approach — we use all sequences for the training of the adaptation model, and we rewrite all phoneme transcripts flowing through the model (i.e., disregarding correctness of the substrings/words). This means, that we create *identity* rewrite rules even for correct words (e.g., "ə → ə") and the n-gram model learns on all this rules.

Our motivation to test this approach is that we assume that the speaker makes errors systematically (e.g., instead of [nt] pronounces [nd] in every word).

Test on artificial data In order to have better control, we first test this naïve method on artificial data. We take the golden phoneme transcripts of the English `read-news-test` and introduce errors to it. We decided to start with one systematic error of the speaker: to pronounce [nt] as [nd] (e.g., "ant" is correctly pronounced [ænt] and would be substituted with [ænd]).

We set the order of the n-gram model to 2 (the smallest allowed order) and simulate the adaptation.

During the first step (first 100 sentences are used for training, following 100 sentences for correction) we dump the obtained rules: We get identity rules for each phoneme except for "t" for which we have two rules ("t $\to$ t" and "t $\to$ d"). This is as expected.

Now we want the n-gram model to "learn" that this rule is applied only for phoneme pair [nd] and in the right words.

We take a closer look at the n-gram training data. We calculate following 2-gram probabilities:

$$P(\text{d} \to \text{d}|\text{n} \to \text{n}) = \frac{c(\text{d} \to \text{d}|\text{n} \to \text{n})}{\sum\limits_{r \in \text{Rules}} c(r|\text{n} \to \text{n})} = \frac{82}{398}, \tag{6.1}$$

$$P(\text{d} \to \text{t}|\text{n} \to \text{n}) = \frac{c(\text{d} \to \text{t}|\text{n} \to \text{n})}{\sum\limits_{r \in \text{Rules}} c(r|\text{n} \to \text{n})} = \frac{61}{398}, \tag{6.2}$$

$$\forall r \in \text{Rules}, r \neq \text{n} \to \text{n} : P(\text{d} \to \text{t}|r) = 0. \tag{6.3}$$

We also check the probabilities given by the KenLM. For both rewrite rules Equations (6.1) and (6.2), the KenLM outputs equal probabilities, which is incorrect. We assume this may be due to lower precision that is needed when working with large amounts of data, which is not our case. To overcome this, we implement our own n-gram model and continue in the analysis.

In the case of extending string "tʃɛk mɛn æn" (Czech men an[d]), the beam search (BS) correctly applies both rules creating two beams. The beam with suffix "d" has a higher probability.

A problem emerges when the BS further extends the beams and comes to point "tʃɛk mɛn ænd wɪmɪn æɹən" (Czech men and women aren'[t]). Again, the BS correctly applies both rules, producing beams "tʃɛk mɛn ænd wɪmɪn æɹənd" and "tʃɛk mɛn ænd wɪmɪn æɹənt". The latter is correct, but the former has higher probability as the 2-gram "n $\to$ n d $\to$ d" has higher probability.

One way, how to minimize the occurrence of this error is to higher the order of the n-gram model. In Figure 6.2, we can observe a clear advantage of the higher-order models over the lower ones.

The naïve approach fails on real data outputted by the acoustic model. We witness the quite substantial deterioration of the performance compared to the non-adapted source in Figure 6.3.

We try to adjust the beam sizes (not in the figure). The bigger the beam, the worse the performance is. The real data contains too irregular errors that seem to be unique each time they occur.

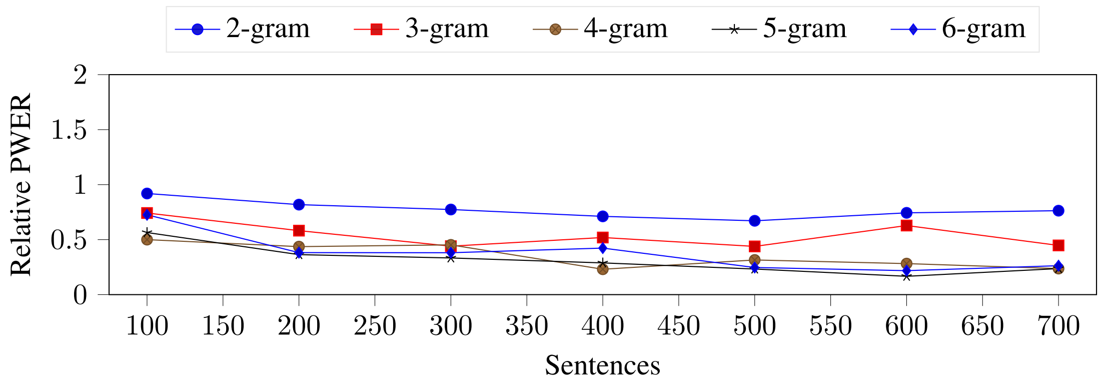


Figure 6.2: Naïve adaptation on artificial data ([nt] replaced with [nd]). PWER of 100 subsequent sentences rewritten using rules obtained from all previous sentences.

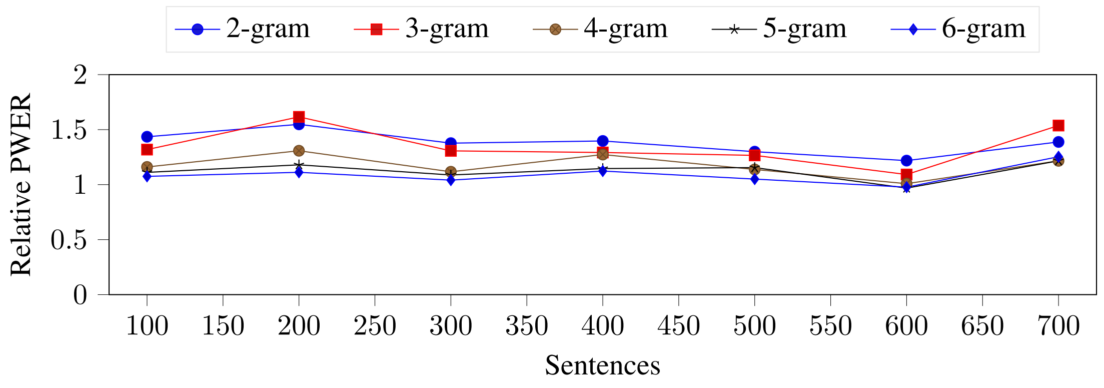


Figure 6.3: Naïve adaptation on English `read-newstest` outputted by the acoustic model. PWER of 100 subsequent sentences rewritten using rules obtained from all previous sentences.

6.2.2 Revised Naïve Approach: Train & Rewrite All Word-by-Word

We revise the naïve approach by splitting the sentences to single words and doing the training and inference word-by-word. Additionally, we use only words differentiating from true transcripts for the learning of the rewriting rules.

On the artificial test set, it seems to perform negligibly better then the previous naïve approach (see Figure 6.4).

For the real acoustic data, the revised method does not bring the desired enhancement and again worsens the PWER relative to the source PWER. It seems to help at the beginning (steps 100 and 200) for 2-gram and 3-gram setups (see Figure 6.5).

As we take a closer look at the evaluation, we observe two together-related problems: Because of the high variability of errors occurring in the real data, there are no “clearly” good rewrite rules that would appear frequently enough. Hence, there are many rules with equal probability based on small frequency. The second problem is that the beam search generates many candidates that are similarly possible due to their similar probability (described in the first problem). However, these beam candidates tend to be contrasting. The beam search is then unable to select the best possible candidate which would be better than others.

This observation leads us to reconsider the configuration of the beam search.

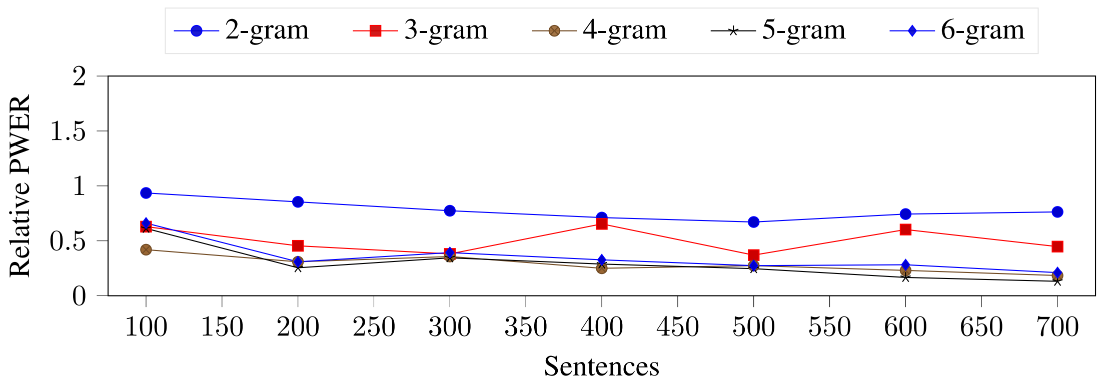


Figure 6.4: Revised naïve adaptation on artificial data ([nt] replaced with [nd]). PWER of 100 subsequent sentences rewritten using rules obtained from all previous sentences.

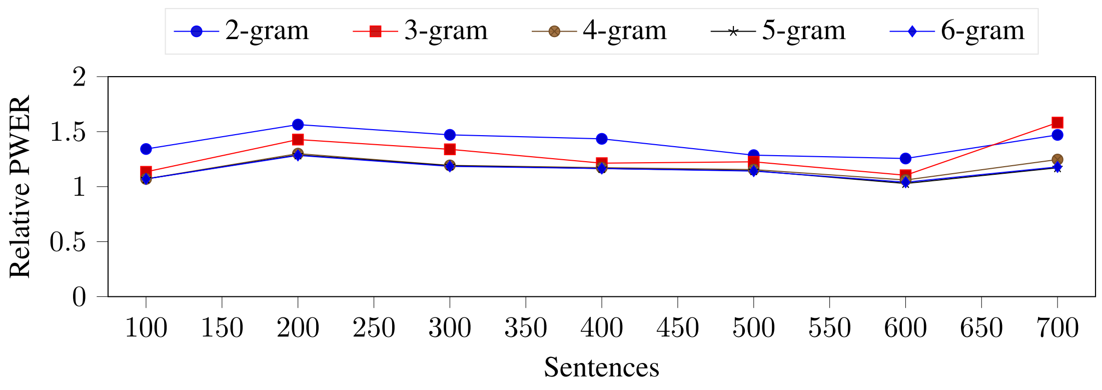


Figure 6.5: Revised naïve adaptation on English `read-newstest` outputted by the acoustic model. PWER of 100 subsequent sentences rewritten using rules obtained from all previous sentences.

6.2.3 Embedded Word Beam Search

The problem of the previously revised approach is that it is unable to select the best-fitting candidate in the context of the sentence. To help the beam search, we propose to use two embedded beam searches operating on different levels. A sentence-level beam search selects appropriate words according to the context and an embedded word-level beam search selecting the best phoneme-rewriting rules.

The embedded beam search applies rewrite rules to one word from the input at the time. This BS matches the BS from the previous naïve approach. However, instead of outputting the most probable beam, it proposes all final beams to the top-level BS.

The top-level beam search uses the candidate words (additionally, we add the original phoneme word from input to the candidates) from the embedded BS to extends its beams. The quality (probability) of a beam is considered using a non-adapting n-gram language model. This could be trained on a large amount of data or, alternatively, for tasks where is the topic or vocabulary known beforehand, one could utilize domain adapted language model.

We first evaluate the artificial data. Figure 6.6 demonstrates the power of this approach. The adaptation learns to fix the error immediately based on the first 100 sentences and applies the rules with almost no errors.

However, on the real data, the new approach does not help (see Figure 6.7).

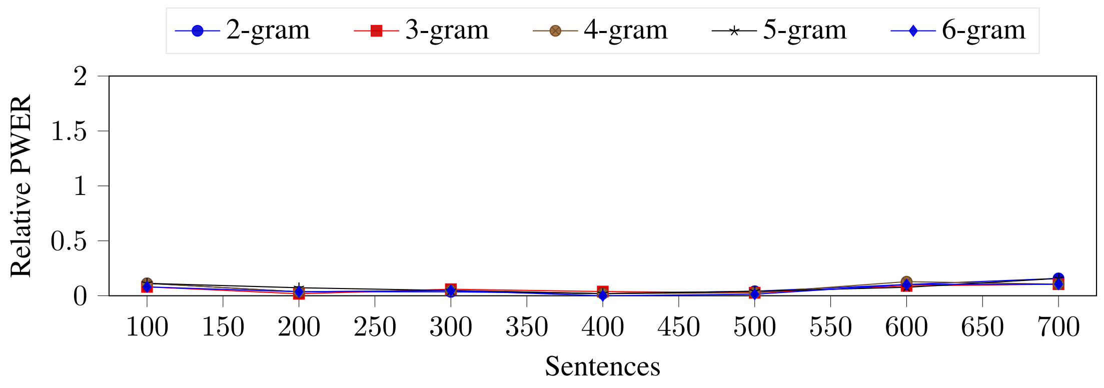


Figure 6.6: Embedded word beam search adaptation on artificial data ([nt] replaced with [nd]). PWER of 100 subsequent sentences rewritten using rules obtained from all previous sentences.

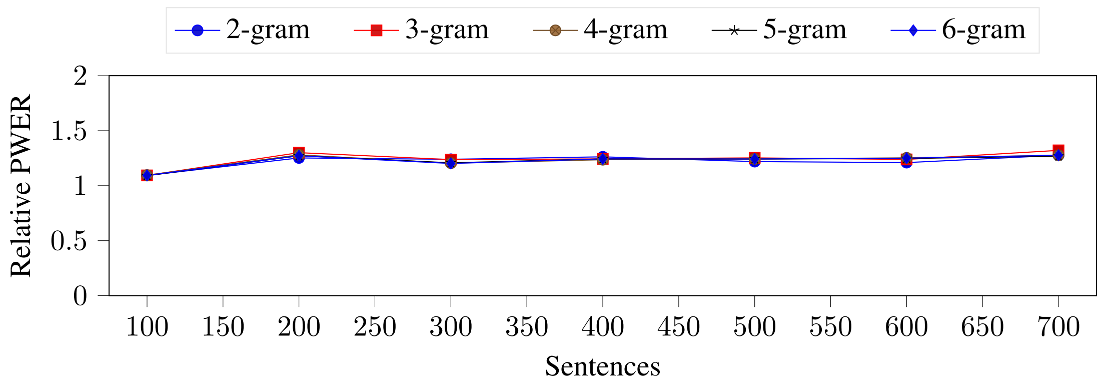


Figure 6.7: Embedded word beam search adaptation on English `read-newstest` outputted by the acoustic model. PWER of 100 subsequent sentences rewritten using rules obtained from all previous sentences.

6.2.4 Analysis

In this section, we analyze the reason why the adaptation model fails to adapt to real acoustic data.

First, we create a new artificial test set with very high PWER by deliberately rewriting the few most commonly used phoneme characters. We guessed from our personal English experience, which phonemes can be mixed-up (e.g., “ð” like in “the [ðə]” to “t” or “d”). We established 18 such rewriting rules. We apply these rewrite rules on each phoneme word independently and with a 60 % probability to make the errors in the data less consistent. We generated data with PWER 54 %. Our best setup, the embedded word beam search, is able to reduce the error to approximately 10 % PWER (notably, a 40 % points reductions).

The main reasons for the failure on real phoneme transcripts may be the vast amount of rewrite rules that are too unique (i.e., have low frequency). In Figure 6.8, we document the number of rules at each step during the adaptation on the English `read-newstest`. We see that the number of rules in the real phoneme transcripts differs at each step with the magnitude of 10.[1] Note, the PWER of the real phoneme

[1]Additionally, we disable the EM alignment algorithm and use the plain modified Levenstein. On the harder artificial data, we observe a substantial reduction of found rules (a half the number) and a further

transcripts is 16 %, and the artificial phoneme transcripts have 54 %.

We also counted how many times each rule occurs in the training data.[2] In Figure 6.9, we show cumulative distribution of rule frequencies (we leave out the identity rewrite rules). We can see that the majority of rules found in real data occurs only once (71 % of rules), 20 % occur two times, 95 % occur at most ten times. On the other hand, the rules with only one occurrence make 28 % percent of all rules in artificial error data. 40 % of all rules are found in the data more than ten times.

Such a vast amount of infrequent rules deteriorates the performance as the search space is too big. The embedded beam search has too many candidates at each step, generating invalid word candidates. The top-level BS then fails to pick the best word as most candidate words are invalid.

We hypothesize that the high number of infrequent rules is caused by substantial variability in speech. Furthermore, we suspect that the neural-network-based acoustic model tends to over-fit to words or sub-words, i.e., the model prefers outputting (in training data) more frequent sub-words rather than best describing the observed speech.

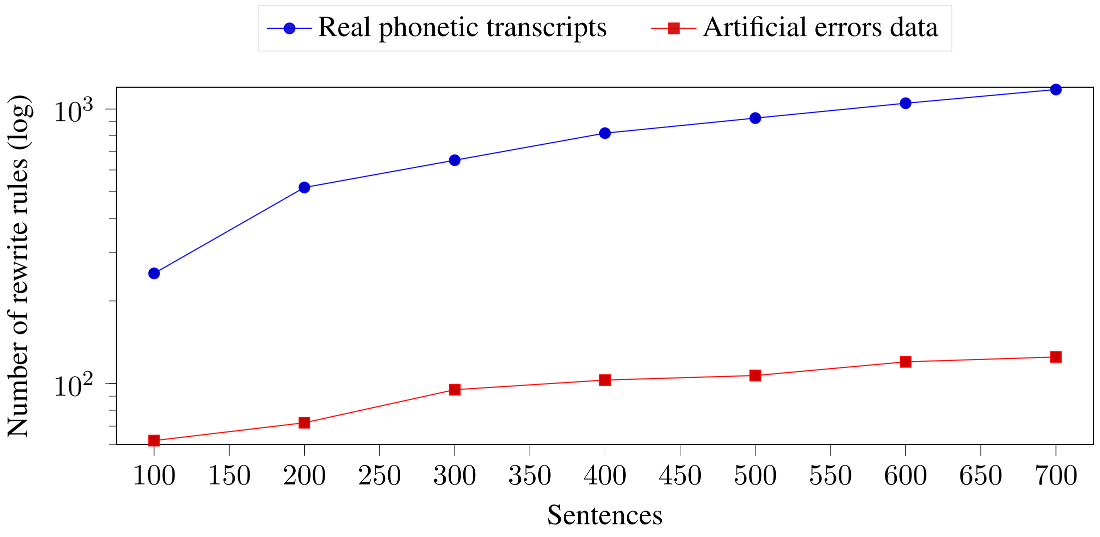


Figure 6.8: Number of rewrite rules found in the acoustic data during the adaptation. We observe substantial difference between the real data and artificial errors data. Note, the real data have 16 % and artificial 54 % PWER.

6.3 Conclusion

We proposed and tested an online speaker adaptation. We demonstrated that the approach works on phoneme transcripts with somewhat regular errors even when the phoneme word error rate is extremely high (we observed PWER reduction of 50 % points of PWER).

On the other hand, our approach fails on real data. We account for this to the extreme irregularity of errors in the real data.

decrease of the PWER (to approximately 4 %). On the other hand, on the real data, this change produces even more rules, which leads to further performance loss.

[2]We attempt to prune rules based on the rule frequency — by a constant (5 and 10) and using percentile (50, 90, and 95). None of the configurations brings enhancement on the real data. In the case of artificial error test set, pruning of 50 % least frequent rules helps further reduce the final PWER.

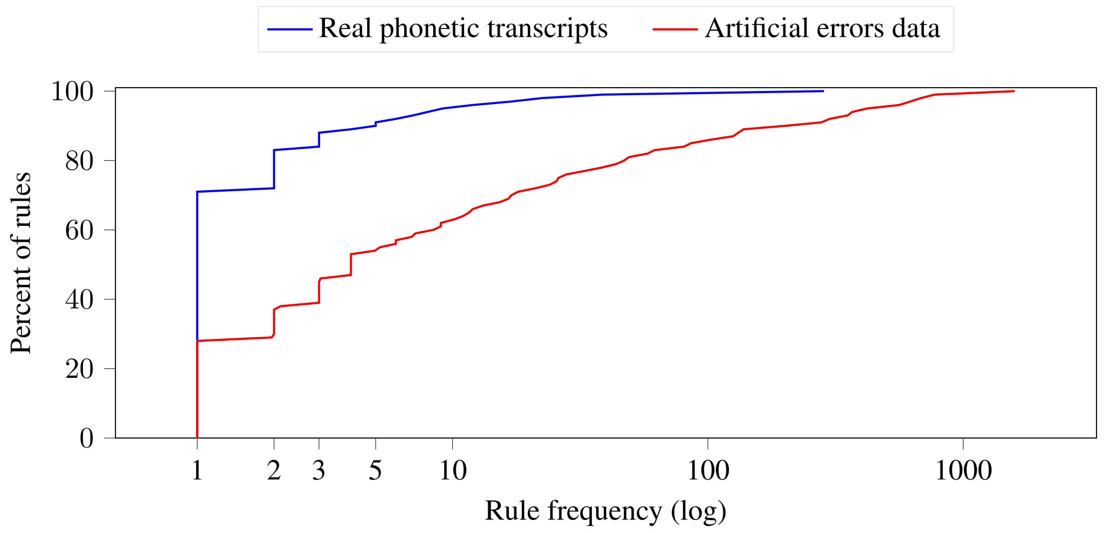


Figure 6.9: Comparison of the rule frequency cumulative distribution on real phonetic transcripts and artificial error data in the last adaptation iteration on English `read-newstest` (rules from 700 sentences). Note, we leave out identity rewrite rules.

We assume that the best possible way for enhancing the performance and robustness of the ASR/SLT is to prepare the translation model for errors during the training.

Conclusion

In this thesis, we explored automatic speech recognition and spoken language translation. More recent studies tend to focus on end-to-end systems, but in this work, we experimented with the more traditional two-step approach of ASR and MT. As opposed to the conventional setup, we use phonemes instead of graphemes as an intermediate representation of speech.

We focus on many details of the proposed approach. First, we examine methods of speeding up training and promoting the final quality of the trained model. We utilize cross-lingual transfer from English to Czech (notably, an unrelated language). To streamline the transfer, we propose our technique — a coarse-to-fine simplification. We proved this method provides a benefit of faster convergence and the better final result compared to both clean Czech training and naïve transfer. Furthermore, this technique also works entirely standalone, outperforming all other setups. We submitted these findings to the Interspeech conference (currently under review).

Further, we build a phoneme acoustic model. To speed up the training, we use transfer from traditional ASR.

An integral part of our ASR/SLT pipeline is the "translation" model. In ASR, this model translates phoneme sequences into grapheme transcripts in the same language. In SLT, the model does the direct translation from the source language phonemes into grapheme translation in the target language.

As our task — the phoneme-to-grapheme translation — is somewhat nontraditional, we experiment with different word segmentation. More precisely, we review the size of the byte pair encoding vocabulary and different sources of training data for the BPE. We found that the best option for a high-resource setting (ours) is the larger vocabulary (32k). The source of training data did not make much difference (we tested clean, corrupt, and mixed setups).

Based on our previous findings, we trained a baseline ASR translation model. We utilized clean training data (phoneme sequences obtained using `phonemizer` tool) without the introduction of any errors. We have proved that this pipeline in the ASR task performs similarly to the end-to-end model Jasper on conventional datasets (LibriSpeech and Common Voice). We observed very promising results on our `read-newstest`, on which our ASR outperforms the end-to-end baseline model. This test set is particularly challenging, as it contains many proper nouns, and a non-native speaker reads the English part. Furthermore, we submitted our ASR system to the International Conference on Spoken Language Translation to the Non-Native Speech Translation track. On the track's development set, our systems outperformed commercial ASR systems by Google and Microsoft (see Polák et al. [2020] for more details).

We also compare the refactored two-step method with the traditional one. We compare the performance of both systems on our own `read-newstest`. For the comparison, we utilize automatic metric SacreBLEU, but we also manually asses the quality. In terms of SacreBLEU, the refactored approach outperforms the baseline. In the manual evaluation, the proposed approach outperforms the baseline in Czech to English and is slightly worse than the baseline in the opposite direction (in the number of correct translations). On the other hand, in both directions, the proposed system has more correct and "almost" correct translations.

Besides establishing baselines, we also experimented with enhancing the pipeline robustness. We review the transfer from SLT and fine-tune the “translation” system on corrupted data, to promote its ability to recover errors introduced in the acoustic model. We were able to reduce the WER on all test sets compared to the baseline.

Finally, we explored adaptation on the fly. We trained an adaptation model that is able to reduce extremely high WER (50 % to 10 %) on artificially corrupted phoneme transcripts. However, we were unable to obtain any enhancement on real transcripts. A probable cause of this is extreme variance in errors in the real data. We think the best possible option, how to recover errors introduced in the acoustic model, is to prepare a robust translation model.

In summary, we are strongly convinced that the refactored two-step approach with phonemes as the intermediate representation has great potential in robust ASR and SLT. Therefore, in the future, we would like to explore this area more thoroughly. Among others, we would like to:

- review and generalize the coarse-to-fine transfer to other languages and alphabets,
- simplify the fine-tuning of ASR, automate the creation of “corrupted” data that could also be used during the training of SLT,
- multi-task learning of ASR and SLT (with shared encoders/decoders),
- “onlinezation” of the whole pipeline with an emphasis on low latency by providing a partial translation of sentences,
- introduce abstract features to transport non-verbal cues from the acoustic to the translation model. This would provide a similar experience as in the end-to-end system, with the convenience of the two-step setup.

Bibliography

Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182, 2016.

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. *arXiv preprint arXiv:1912.06670*, 2019.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949. IEEE, 2016.

Loïc Barrault, Ondřej Bojar, Marta R Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, et al. Findings of the 2019 conference on machine translation (wmt19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, 2019.

Willem D Basson and Marelie H Davel. Category-based phoneme-to-grapheme transliteration. 2013.

Alexandre Bérard, Olivier Pietquin, Christophe Servan, and Laurent Besacier. Listen and translate: A proof of concept for end-to-end speech-to-text translation. *arXiv preprint arXiv:1612.01744*, 2016.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. End-to-end automatic speech translation of audiobooks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6224–6228. IEEE, 2018.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovickỳ, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. Czeng 1.6: enlarged czech-english parallel corpus with processing tools dockered. In *International Conference on Text, Speech, and Dialogue*, pages 231–238. Springer, 2016a.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, et al. Proceedings of the third conference on machine translation: Shared task papers. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, 2018.

Ondřej Bojar, Ondřej Cífka, Jindřich Helcl, Tom Kocmi, and Roman Sudarikov. Ufal submissions to the iwslt 2016 mt track. In *Proceedings of the ninth International Workshop on Spoken Language Translation (IWSLT)*, pages 1–8. Karlsruhe Institute of Technology, 2016b.

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. Multilevel coarse-to-fine PCFG parsing. In Robert C. Moore, Jeff A. Bilmes, Jennifer Chu-Carroll, and Mark Sanderson, editors, *HLT-NAACL*. The Association for Computational Linguistics, 2006. URL `http://acl.ldc.upenn.edu/N/N06/N06-1022.pdf`.

Dongpeng Chen, Brian Mak, Cheung-Chi Leung, and Sunil Sivadas. Joint acoustic modeling of triphones and trigraphemes by multi-task learning deep neural networks for low-resource speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5592–5596. IEEE, 2014.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, 2018.

Jaejin Cho, Murali Karthick Baskar, Ruizhi Li, Matthew Wiesner, Sri Harish Mallidi, Nelson Yalta, Martin Karafiat, Shinji Watanabe, and Takaaki Hori. Multilingual sequence-to-sequence speech recognition: architecture, transfer learning, and language modeling. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 521–527. IEEE, 2018.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. End-to-end continuous speech recognition using attention-based recurrent nn: First results. *arXiv preprint arXiv:1412.1602*, 2014.

Walter Daelemans, Antal Van Den Bosch, and Ton Weijters. Igtree: using trees for compression and classification in lazy learning algorithms. In *Lazy learning*, pages 407–423. Springer, 1997.

Walter Daelemans, Jakub Zavrel, Kurt Van Der Sloot, and Antal Van den Bosch. Timbl: Tilburg memory-based learner. *Tilburg University*, 2004.

B. Decadt, J. Duchateau, W. Daelemans, and P. Wambacq. Phoneme-to-grapheme conversion for out-of-vocabulary words in large vocabulary speech recognition. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2001. ASRU '01.*, pages 413–416, 2001.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Michael Denkowski and Graham Neubig. Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Shuoyang Ding, Adithya Renduchintala, and Kevin Duh. A call for prudent choice of subword merge operations in neural machine translation. In *Proceedings of Machine Translation Summit XVII Volume 1: Research Track*, pages 204–213, 2019.

Li Dong and Mirella Lapata. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1068. URL `https://www.aclweb.org/anthology/P18-1068`.

Jonathan G Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 347–354. IEEE, 1997.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

Rohit Gupta, Laurent Besacier, Marc Dymetman, and Matthias Gallé. Character-based nmt with transformer. *arXiv preprint arXiv:1911.04997*, 2019.

Kyu J Han, Ramon Prieto, Kaixing Wu, and Tao Ma. State-of-the-art speech recognition using multi-stream self-attention with dilated 1d convolutions. *arXiv preprint arXiv:1910.00716*, 2019.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

Kenneth Heafield. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July 2011. URL `https://kheafield.com/papers/avenue/kenlm.pdf`.

Axel Horndasch, Elmar Nöth, Anton Batliner, and Volker Warnke. Phoneme-to-grapheme mapping for spoken inquiries to the semantic web. In *Ninth International Conference on Spoken Language Processing*, 2006.

Oleksii Hrinchuk, Mariya Popova, and Boris Ginsburg. Correction of automatic speech recognition with transformer sequence-to-sequence model. *arXiv preprint arXiv:1910.10697*, 2019.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Ye Jia, Melvin Johnson, Wolfgang Macherey, Ron J Weiss, Yuan Cao, Chung-Cheng Chiu, Naveen Ari, Stella Laurenzo, and Yonghui Wu. Leveraging weakly supervised data to improve end-to-end speech-to-text translation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7180–7184. IEEE, 2019.

Biing-Hwang Juang and Lawrence R Rabiner. Automatic speech recognition–a brief history of the technology development. *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, 1:67, 2005.

Uday Kamath, John Liu, and James Whitaker. *Deep learning for nlp and speech recognition*. Springer, 2019.

Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al. A comparative study on transformer vs rnn in speech applications. In *Proceedings of the ASRU 2019 IEEE Automatic Speech Recognition and Understanding Workshop*, 2019. (in print).

S. Kim and M. L. Seltzer. Towards language-universal end-to-end speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4914–4918, April 2018. doi: 10.1109/ICASSP.2018.8462201.

Tom Kocmi and Ondřej Bojar. Trivial transfer learning for low-resource neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 244–252, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6325. URL `https://www.aclweb.org/anthology/W18-6325`.

Jonáš Kratochvíl, Peter Polák, and Ondřej Bojar. Large Corpus of Czech Parliament Plenary Hearings. `http://hdl.handle.net/11234/1-3126`, 2019.

Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, and Yang Zhang. Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions. *arXiv preprint arXiv:1910.10261*, 2019.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, 2018.

Julius Kunze, Louis Kirsch, Ilia Kurenkov, Andreas Krug, Jens Johannsmeier, and Sebastian Stober. Transfer learning for speech recognition on a budget. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 168–177, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2620. URL `https://www.aclweb.org/anthology/W17-2620`.

Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M. Cohen, Huyen Nguyen, and Ravi Teja Gadde. Jasper: An End-to-End Convolutional Neural Acoustic Model. In *Proc. Interspeech 2019*, pages 71–75, 2019a. doi: 10.21437/Interspeech.2019-1819. URL `http://dx.doi.org/10.21437/Interspeech.2019-1819`.

Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M Cohen, Huyen Nguyen, and Ravi Teja Gadde. Jasper: An end-to-end convolutional neural acoustic model. *arXiv preprint arXiv:1904.03288*, 2019b.

David Lubensky. Learning spectral-temporal dependencies using connectionist networks. In *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, pages 418–421. IEEE, 1988.

Minh-Thang Luong and Christopher D Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, 2016.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*, 2014.

V. Moshkelgosha, H. Behzadi-Khormouji, and M. Yazdian-Dehkordi. Coarse-to-fine parameter tuning for content-based object categorization. In *2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA)*, pages 160–165, April 2017. doi: 10.1109/PRIA.2017.7983038.

Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *arXiv preprint arXiv:1003.4083*, 2010.

Jayashree Padmanabhan and Melvin Jose Johnson Premkumar. Machine learning in automatic speech recognition: A survey. *IETE Technical Review*, 32(4):240–251, 2015.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

Douglas B Paul and Janet M Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*, pages 357–362. Association for Computational Linguistics, 1992.

Thuong-Hai Pham, Dominik Macháček, and Ondřej Bojar. Promoting the knowledge of source syntax in transformer nmt is not needed. *Computación y Sistemas*, 23(3): 923–934, 2019. ISSN 1405-5546.

Peter Polák et al. Neural ASR with Phoneme-Level Intermediate Step – A Non-Native Speech Translation Task Submission to IWSLT 2020. In *The International Workshop on Spoken Language Translation (IWSLT)*, TBD, TBD 2020.

Martin Popel and Ondřej Bojar. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70, 2018.

Matt Post. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, 2018.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. Bpe-dropout: Simple and effective subword regularization. *arXiv preprint arXiv:1910.13267*, 2019.

C. Raphael. Coarse-to-fine dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1379–1390, Dec 2001. ISSN 1939-3539. doi: 10.1109/34.977562.

D. Reddy and A. Robinson. Phoneme-to-grapheme translation of english. *IEEE Transactions on Audio and Electroacoustics*, 16(2):240–246, 1968.

Michael D Riley and Andrej Ljolje. Recognizing phonemes vs. recognizing phones: a comparison. In *Second International Conference on Spoken Language Processing*, 1992.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Elizabeth Salesky, Matthias Sperber, and Alan W Black. Exploring phoneme-level speech representations for end-to-end speech translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1835–1841, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1179. URL `https://www.aclweb.org/anthology/P19-1179`.

Michael L Seltzer and Jasha Droppo. Multi-task learning in deep neural networks for improved phoneme recognition. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6965–6969. IEEE, 2013.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. Attention-passing models for robust and data-efficient end-to-end speech translation. *Transactions of the Association for Computational Linguistics*, 7:313–325, 2019.

Stanley Smith Stevens, John Volkmann, and Edwin B Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937.

Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International Conference on Artificial Neural Networks*, pages 270–279. Springer, 2018.

Jian Tang, Yan Song, Lirong Dai, and Ian McLoughlin. Acoustic modeling with densely connected residual network for multichannel speech recognition. *Proc. Interspeech 2018*, pages 1783–1787, 2018.

Sibo Tong, Philip N. Garner, and Hervé Bourlard. Cross-lingual adaptation of a ctc-based multilingual acoustic model. *Speech Communication*, 104:39 – 46, 2018. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2018.09.001. URL `http://www.sciencedirect.com/science/article/pii/S016763931830030X`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

Ron J Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, and Zhifeng Chen. Sequence-to-sequence models can directly translate foreign speech. *Proc. Interspeech 2017*, pages 2625–2629, 2017.

Dong Yu and Li Deng. *AUTOMATIC SPEECH RECOGNITION.* Springer, 2016.

Ying Zhang, Mohammad Pezeshki, Philémon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, and Aaron Courville. Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*, 2017.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. Coarse-to-fine learning for neural machine translation. In Min Zhang, Vincent Ng, Dongyan Zhao, Sujian Li, and Hongying Zan, editors, *Natural Language Processing and Chinese Computing*, pages 316–328, Cham, 2018. Springer International Publishing. ISBN 978-3-319-99495-6.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1163. URL `https://www.aclweb.org/anthology/D16-1163`.

List of Figures

1.1 Jasper Dense Residual architecture 9
1.2 QuartzNet BxR architecture 10
1.3 Convolutional module with groups and shuffle 11
1.4 Transformer model architecture 12
1.5 Amplitude and mel spectograms of “Hello World!”. 14

2.1 Examined setups of transfer learning 21
2.2 Evaluation on test set during CS ASR training 24
2.3 Learning curves for Czech acoustic model 32
2.4 Learning curves for English acoustic model 33

3.1 Enhanced ASR pipeline 35
3.2 Training captured on LibriSpeech corrupted `dev-clean` 39
3.3 Training captured on Common Voice corrupted `dev` set 41
3.4 Results on the Common Voice `test` set 41
3.5 Results on the LibriSpeech `test-clean` 42
3.6 Results on the LibriSpeech `test-other` 42
3.7 Czech ASR training captured on `newstest2015` 45
3.8 English ASR training captured on `newstest2015` 46

4.1 Czech to English training 50
4.2 English to Czech training 51

5.1 Overview of setup for obtaining “corrupted” training data. 61
5.2 English ensemble training 62
5.3 PWER distribution of collected English data 63
5.4 Czech ensemble training 64
5.5 PWER distribution of collected Czech data 65
5.6 English enhanced ASR fine-tuning 67
5.7 Czech enhanced ASR fine-tuning 68

6.1 Enhanced ASR pipeline with online adaptation. 71
6.2 Naïve adaptation on artificial data 74
6.3 Naïve adaptation on English 74
6.4 Revised naïve adaptation on artificial data 75
6.5 Revised naïve adaptation on English 75
6.6 Embedded word beam search adaptation on artificial data 76
6.7 Embedded word beam search adaptation on English 76
6.8 Number of rewrite rules found in the acoustic data during the adaptation 77
6.9 Comparison of the rule frequency cumulative distribution 78

List of Tables

2.1 Results in % of WER on the Czech (CS) test set 23
2.2 The best Czech model performance 26
2.3 Examples of multi-character phonemes 31
2.4 Results of Czech acoustic models 32
2.5 Results of English acoustic models 33

3.1 Results on the Common Voice `dev` set 39
3.2 Results on the LibriSpeech 39
3.3 Results on the Common Voice `test` set 41
3.4 Results on the LibriSpeech `test-clean` 42
3.5 Results on the LibriSpeech `test-other` 42
3.6 Czech enhanced ASR results 44
3.7 English enhanced ASR results 45

4.1 Acoustic models performance on `read-newstest` 51
4.2 Czech to English manual evaluation 53
4.3 Samples of names from Czech to English SLT. 53
4.4 Sentence samples from Czech to English SLT. 54
4.5 Czech to English SLT evaluation 55
4.6 English to Czech manual evaluation 56
4.7 Samples of names from English to Czech SLT. 57
4.8 Sentence samples from English to Czech SLT. 57
4.9 English to Czech SLT evaluation 58

5.1 Final English ensemble performance 62
5.2 Performance of the 40 ensemble models on the datasets. 63
5.3 Czech ensemble final performance 64
5.4 English finetuned ASR performance evaluation 69
5.5 Czech finetuned ASR performance evaluation 69

List of Abbreviations

ANN	artificial neural network
ASR	automatic speech recognition
AVG	average, mean
BLEU	translation quality metric; bilingual evaluation understudy
BPE	byte pair encoding
BS	beam search
CS	Czech
CTC	connectionist temporal classification
DNN	deep neural
E2E	end-to-end
EN	English
GMM	gaussian mixture model
HMM	hidden Markov model
K	kilo, thousand
LM	language model
M	mega, million
MFCC	mel-frequency cepstral coefficients
MT, NMT	(neural) machine translation
OOV	out-of-vocabulary
P2G	phoneme-to-grapheme
PWER	phoneme word error rate
SLT	spoken language translation
STD	standard deviation
WER	word error rate

A. Attachments

A.1 Electronic Attachment Contents

For easier replication of our results and analysis, we also attach some scripts and outputs of used models and algorithms. Note, we do not include models, datasets nor all outputs due to the limited space.

The following is included:

1. `thesis.pdf` — this thesis in PDF format (as uploaded to the SIS),
2. iwslt2020.pdf — paper accepted to IWSLT2020 [Polák et al., 2020].
3. interspeech2020.pdf – paper submitted to Interspeech 2020,
4. `acoustc_models/` — scripts and outputs of the acoustic and ASR models as discussed in Chapter 2,
5. `translation_models/` — the Transformer models, ASR transcripts (described in Chapters 3 and 5) and SLT translations (discussed in Chapter 4),
6. `online_adaptation/` — the online adaptation algorithms (described in Chapter 6), test sets and outputs.