



**Zurich University of Applied Sciences**

School of Engineering  
Centre for Artificial Intelligence

---

# **Enhancing Error-Preserving Automatic Speech Recognition of Young English Learners' Language Through Pipeline-Based Speech Data Augmentation**

---

A thesis by

Janick Michot

for the degree of

Master of Science

Supervised by

Prof. Dr. Mark Cieliebak

Dr. Jan Milan Deriu

January 31, 2025

# Abstract

Speaking is a central yet underemphasized skill in second-language learning, especially for children who have limited speaking opportunities in school. Chatbots can offer interactive speaking opportunities, but they rely on Automated Speech Recognition (ASR) systems that accurately transcribe spontaneous child speech. Most ASR models, however, are trained on scripted speech from native adults, making them ineffective for transcribing the spontaneous, error-prone speech of young learners. Additionally, most ASR systems contain a language model that automatically corrects errors, limiting their ability to capture learner mistakes essential for corrective feedback. To develop an ASR system that works on spontaneous speech by young language learners and preserves their mistakes, targeted data is required. But collecting child speech data is costly and time-consuming. Thus, this work investigates the impact of synthetic speech data on ASR accuracy and error preservation for young Swiss learners of English.

For this, a multistep data augmentation pipeline was developed to generate child-like speech. An LLM-based text sample generator produced 34,000 parallel sentences with learner-specific errors. Using the F5 TTS with voice cloning capabilities and the generated text samples, a 100-hour synthetic speech corpus was created by synthesizing speech based on real children's voices. This dataset augmented real children's speech and was integrated into mixed-training experiments with varying real-to-synthetic data ratios to fine-tune ASR models. Results show that synthetic speech enhances transcription accuracy and error preservation, with the best configuration reducing Word Error Rate (WER) by 1.19% and Word-Based Error Preservation Rate (WEPR) by 2.73%, though effectiveness depends on proportions of real and synthetic data. Ultimately, this study demonstrates the potential of speech synthesis while underscoring the need for further refinement. Improving the pipeline at multiple stages is essential to enhance speech quality and error preservation. These refinements could lead to more robust and specialized ASR systems, thereby contributing to more effective second-language acquisition.

# Contents

<b>Abstract</b>	ii
<b>Contents</b>	iii
<b>List of Figures</b>	vi
<b>List of Tables</b>	vii
<b>List of Abbreviations</b>	viii
<b>1 Introduction</b>	1
1.1 Background and Context . . . . .	2
1.2 Problem Statement . . . . .	3
1.3 Research Objectives . . . . .	4
1.4 Key Findings & Contributions . . . . .	5
1.5 Structure of the Thesis . . . . .	6
<b>2 Fundamentals</b>	7
2.1 Large Language Models . . . . .	7
2.1.1 GPT . . . . .	9
2.1.2 Prompting . . . . .	9
2.1.3 In-Context Learning (ICL) . . . . .	10
2.1.4 Prompt Chaining . . . . .	10
2.2 Text-to-Speech . . . . .	11
2.2.1 State-of-the-Art TTS . . . . .	12
2.2.2 Voice Cloning . . . . .	12
2.2.3 E2-TTS . . . . .	13
2.2.4 F5-TTS . . . . .	13
2.2.5 Coqui XTTS . . . . .	14
2.2.6 TTS Evaluation . . . . .	14
2.2.7 Similarity Mean Opinion Score (SMOS) . . . . .	14
2.2.8 Comparative Mean Opinion Score (CMOS) . . . . .	14
2.2.9 DataSpeech . . . . .	15
2.3 Automated Speech Recognition (ASR) . . . . .	16
2.3.1 wav2vec 2.0 . . . . .	18
2.3.2 Whisper . . . . .	20
2.3.3 Evaluation Metrics for ASR . . . . .	20
2.4 Grammatical Error Correction . . . . .	22
2.4.1 ERRANT . . . . .	22
<b>3 Related Work</b>	24
3.1 Children's Speech Corpora. . . . .	24
3.2 ASR for children's speech and language learners. . . . .	25
3.3 Data Augmentation for Children's Speech and Language Learners . . . . .	26
3.4 Data Augmentation for Erroneous Texts . . . . .	28

<b>4 Pipeline-Based Experiment Setup</b>	<b>29</b>
4.1 Pipeline Framework Overview . . . . .	29
4.1.1 ChaLL Data . . . . .	31
4.1.2 Human-Annotated Subset with ERRANT Errors . . . . .	32
4.2 Modular Service-Based Framework . . . . .	32
4.3 Experiment Execution & Tracking . . . . .	34
<b>5 Text Sample Generation</b>	<b>37</b>
5.1 Methods . . . . .	37
5.1.1 Generating Text Pairs . . . . .	38
5.1.2 Sample Generation Chain . . . . .	38
5.1.3 Dataset Generation . . . . .	41
5.2 Results and Deliverables . . . . .	42
5.2.1 ERRANT-annotated ChaLL Subset Evaluation . . . . .	42
5.2.2 Error Type Selection and Definition . . . . .	43
5.2.3 Sample Generation App . . . . .	45
5.2.4 Synthetic Text Dataset . . . . .	46
5.3 Conclusion And Future Work . . . . .	50
<b>6 Audio Generation</b>	<b>52</b>
6.1 Methods . . . . .	52
6.1.1 Definition of Voice References . . . . .	52
6.1.2 TTS System Comparison . . . . .	53
6.1.3 Automated Similarity and Naturalness Evaluation . . . . .	55
6.1.4 Audio Dataset Generation . . . . .	55
6.2 Results and Deliverables . . . . .	57
6.2.1 Code-Switching Findings . . . . .	57
6.2.2 Gradio App for Evaluation . . . . .	57
6.2.3 SMOS and CMOS Evaluation Results . . . . .	58
6.2.4 DataSpeech Evaluation Results . . . . .	59
6.2.5 Synthetic Audio Dataset . . . . .	60
6.3 Conclusion and Future Work . . . . .	60
<b>7 Train ASR with Real and Synthetic Data</b>	<b>62</b>
7.1 Methods . . . . .	62
7.1.1 Data Preparation and Partitioning . . . . .	62
7.1.2 Training and Evaluation Strategy . . . . .	64
7.1.3 ERRANT Error Type Evaluation . . . . .	65
7.2 Results and Deliverables . . . . .	68
7.2.1 Evaluation on Real Data . . . . .	68
7.2.2 Evaluation on Synthetic Data . . . . .	69
7.3 Conclusion and Future Work . . . . .	70
<b>8 Conclusion &amp; Future Work</b>	<b>72</b>
8.1 Conclusion . . . . .	72
8.2 Future Work . . . . .	74

<b>APPENDIX</b>	<b>78</b>
<b>A Sample Generation</b>	<b>79</b>
A.1 ERRANT . . . . .	79
A.2 Sample Generation Algorithm . . . . .	80
A.3 Evaluation of Annotated Subset . . . . .	81
A.4 Sample Generation Prompts . . . . .	82
A.4.1 Subject Generation . . . . .	82
A.4.2 Grammar Generation . . . . .	82
A.4.3 Prompt Manager . . . . .	83
A.5 Sample Generation Sequence Inputs . . . . .	87
A.6 Sample Generation App . . . . .	88
A.7 Generated Texts Dataset Evaluation . . . . .	89
<b>B Audio Generation</b>	<b>93</b>
B.1 Evaluation App . . . . .	93
B.2 TTS System Evaluation . . . . .	94
B.3 Dataspeech Evaluation . . . . .	95
<b>C Training Appendix</b>	<b>96</b>
C.1 Performance of Human vs. Automatic ERRANT Error Detection . . . . .	96
C.2 Error Type Evaluation on Synth Data . . . . .	97
C.3 Error Type Evaluation on Real Data . . . . .	98
<b>Declarations</b>	<b>99</b>
<b>Bibliography</b>	<b>100</b>

# List of Figures

1.1 Overview of the Methodology: Text Sample Generation, Audio Dataset Generation, and ASR Training . . . . .	5
4.1 Extented Overview of the Methodology: Text Generation, Audio Dataset Generation, and Mixed Data Training Workflow . . . . .	30
4.2 Pipeline Service and Execution ERD . . . . .	33
4.3 Functionality and Structure of a Launch Job . . . . .	35
5.1 Parallel Text Sample Generation Framework using Prompt Chaining . . . . .	39
7.1 Distribution of School Grades Across Splits . . . . .	63
A.1 ERRANT Error Type Distribution in the Annotated Subset . . . . .	81
A.2 Screenshot of the Sample Generation App . . . . .	88
A.3 Generated Text Dataset Evaluation: Distribution of Code-Switching Patterns and Text Properties . . . . .	89
A.4 Generated Text Dataset Evaluation: Target Error Counts vs. ERRANT Detected Error Counts . . . . .	89
A.5 Generated Text Dataset Evaluation: Distribution of Configured and Detected Errors . . . . .	91
A.6 Generated Text Dataset Evaluation: Confusion Matrix of Configured vs. Detected Errors	92
B.1 Screenshot of Sample Generation App Interface . . . . .	93
B.2 TTS System Evaluation: SMOS and CMOS distributions. . . . .	94
B.3 Dataspeech Evaluation: Alignment of Audio Features . . . . .	95

# List of Tables

2.1 Sample Alignment between an Original and Corrected sentence . . . . .	22
5.1 Human Annotated ERRANT Errors: Error Distribution Across Linguistic Tiers . . . . .	42
5.2 Human Annotated ERRANT Errors: Examples of Frequent Error Types . . . . .	43
5.3 Overview of Selected Dataset Samples from Synthethic Text Dataset . . . . .	47
5.4 Performance Metrics for Text Dataset Evaluation . . . . .	50
6.1 Evaluation Results for Different TTS Systems (SMOS and CMOS) . . . . .	58
6.2 Linear Regression Analysis of Selected TTS Systems . . . . .	59
7.1 Training Data Composition Grid . . . . .	64
7.2 Error Detection: Samples for Simple Error Detection . . . . .	66
7.3 Error Detection: Samples for Multiple Corrections Union . . . . .	66
7.4 Error Detection: Samples for Multiple Corrections Error Majority . . . . .	67
7.5 Error Detection: Samples for Multiple Corrections Span Majority . . . . .	67
7.6 Error Detection: Samples for Multiple Corrections Probabilistic . . . . .	68

7.7	WER Evaluation on Real Test Data . . . . .	68
7.8	WEPR Evaluation on Real Test Data . . . . .	68
7.9	WER Evaluation on Synthetic Test Data . . . . .	69
A.1	ERRANT Error Type Overview . . . . .	79
A.2	Summary of ERRANT Error Codes Used for Generation . . . . .	87
A.3	Comparison of Configured vs. Detected Errors . . . . .	90
C.1	Performance of Human vs. Automatic ERRANT Error Detection . . . . .	96
C.2	Error Type Evaluation on Synth Data: Comparing multiple Error Detection Strategies. .	97
C.3	Error Type Evaluation on Real Data: Comparing multiple Error Detection Strategies. .	98

# List of Abbreviations

- AED** Attention-based Encoder-Decoder. 18
- AI** Artificial Intelligenc. 7, 9
- AR** Autoregressive. 12
- ASR** Automated Speech Recognition. ii–iv, vi, 2–7, 11, 16, 17, 19–22, 24, 26–29, 31, 38, 62, 64–66, 68–77
- BLEU** Bilingual Evaluation Understudy. 20
- CER** Character Error Rate. 21, 27, 61
- ChaLL** Chatbot for Language Learners. 2–5, 8, 25, 26, 29, 31, 32, 37, 38, 41, 51–54, 56, 62, 63, 65, 72, 77
- chrF** Character Level F-Score. 21
- CMOS** Comparative Mean Opinion Score. iii, 4, 6, 14, 31, 54, 55, 57–59, 93, 94
- CNN** Convolutional Neural Network. 12
- CTC** Connectionist Temporal Classification. 17, 18
- DNN** Deep Neural Network. 11
- E2E** End-to-End. 16–18
- GEC** Grammatical Error Correction. 7, 22, 28, 31, 38, 65–68, 70, 71, 77
- HMM** Hidden Markov Model. 11
- ICL** In-Context Learning. iii, 10
- L2** Second Language. 2, 3, 24, 28
- LLM** Large Language Model. ii, 4–10, 24, 27, 28, 37–41, 50, 51, 72
- LSTM** Long Short-Term Memory. 11, 12
- NAR** Non-Autoregressive. 12–14
- PLM** Pre-Trained Language Model. 7
- POS** Part-of-Speech. 22, 23
- RNN** Recurrent Neural Network. 11, 12
- SMOS** Similarity Mean Opinion Score. iii, 4, 6, 14, 31, 54, 55, 57–59, 93, 94
- SOTA** State of the art. 19, 20
- SSL** Self-Supervised Learning. 16
- TTS** Speech-to-Text. 5, 16, 20, 70
- TTS** Text-to-Speech. iii, iv, 3–7, 11–16, 27–29, 44, 45, 52–58, 60, 61, 70, 73, 75–77
- W&B** Weights and Biases. 34–36
- WEPR** Word-Based Error Preservation Rate. ii, 3, 21, 22, 26, 31, 62, 65, 68–70, 72, 73
- WER** Word Error Rate. ii, 3, 20, 21, 26–28, 31, 38, 61, 65, 68–70, 72–74

# 1

## Introduction

Speaking is a fundamental competency in foreign language education and is the second most used skill in everyday communication (Hedge, 2001). It is vital for learners, fostering linguistic proficiency, confidence, and practical communication competence, allowing them to express their thoughts, emotions, and ideas clearly and persuasively. Effective communication through speaking is a critical skill for personal, academic, and professional success, particularly in English, which has become the lingua franca of global communication (Rao, 2019). However, speaking often receives less focus compared to reading and writing in many educational settings (Pakula, 2019).

But speaking is inherently tied to the development of other language skills. Burns (2019) highlights that speaking activities can enhance listening skills, foster vocabulary acquisition, and improve grammatical accuracy when carefully integrated into the curriculum. By engaging in meaningful spoken interactions, learners also develop cognitive and metacognitive strategies that aid their overall language proficiency.

Despite its importance, teaching speaking remains a challenge for many educators due to factors such as limited class time, emphasis on written skills, and lack of pedagogical content knowledge (Pakula, 2019). Teaching speaking involves more than just enabling students to “do speaking” through activities; it requires structured, explicit instruction that addresses linguistic features, speech strategies, and the sociocultural contexts of communication (Burns, 2019). In language education, speaking serves as both a medium for learning and an outcome to be achieved (Burns, 2019).

To effectively develop speaking skills, students need to be systematically trained from the early stages of second language acquisition. However, speech production is a complex process requiring the integration of linguistic, cognitive, and social skills in real time (Burns, 2019) that is often not addressed adequately in classrooms. The primary problem is that students have limited access to speaking opportunities (Grimm et al., 2015) and insufficient engagement in extended conversational practice (Pfenninger & Lendl, 2017). Further, learners often face psychological barriers such as anxiety and lack of confidence, which hinder their ability to practice and improve speaking skills in class (Pakula, 2019).

Recent progress in speech processing (Malik et al., 2021) and conversational dialogue systems (Deriu et al., 2021; Ni et al., 2022) presents an opportunity to enhance language learners’ speaking practice through automated tools.

1.1 Background and Context	2
1.2 Problem Statement . . .	3
1.3 Research Objectives . . .	4
1.4 Key Findings & Contributions . . . . .	5
1.5 Structure of the Thesis .	6

## 1.1 Background and Context

The work presented in this thesis is part of a larger effort to develop an interactive, voice-driven Chatbot for Language Learners (ChaLL) that allows children to practice and improve their conversational speaking abilities (Hürlimann et al., 2024). The chatbot acts as a virtual speaking companion, tailored to align with the learners' language proficiency and interests, while providing corrective feedback to support their progress in language acquisition. In both open-ended conversations and structured tasks, the bot enables children to build essential speech functions for real-life activities.

One key challenge in this process is the Automated Speech Recognition (ASR) system, which transcribes learners' spoken input into text for further processing in downstream tasks such as identifying speaking errors, managing dialogue interactions, and providing feedback. In this context, the challenge is not just to transcribe the speech accurately but also to capture the learners' mistakes. This is vital for providing effective corrective feedback, which helps learners recognize their mistakes and encourages them to revise their speech, ultimately fostering language development (Ellis, 2021). But current state-of-the-art ASR models tend to correct the speakers' errors, making corrective feedback impossible. In a previous work (Michot et al., 2024) it was showed that these systems often fail to capture non-standard features such as learner-specific errors and code-switching commonly produced by Swiss children aged 9–14. This limitation affects the chatbot's ability to provide accurate responses and effective corrective feedback.

Another significant challenge for ASR systems is handling spontaneous children's speech. Most of these systems are trained on adult read-aloud error-free corpora recorded by native speakers (Ardila et al., 2020; Panayotov et al., 2015). But children's speech, particularly spontaneous speech by language learners, differs significantly from the read-aloud speech of native adult speakers. These differences arise from their ongoing physical and linguistic development, which influences both their voice characteristics (spectral and temporal) and pronunciation (Gurunath Shivakumar & Georgiou, 2020). This difference arises primarily from the developmental changes in their vocal tract, influencing sound frequencies (spectral content) and vocal resonance (Potamianos & Narayanan, 2003). Additionally, children's speech exhibits high variability both within individuals over time (Gerosa et al., 2006) and across age groups (inter-speaker variability) (Lee et al., 1999a). Children's speech shows greater variability between spontaneous and read-aloud than adult speech, making it even more challenging for models trained on standard adult speech corpora (Gerosa et al., 2009).

Transcribing the English of Swiss learners adds another layer of complexity due to the characteristics of L2 speech. L2 learners' speech is influenced by interlanguage (a mix of their native language and English) which introduces unpredictable grammar,

vocabulary, and pronunciation patterns (Selinker, 1972). Unlike native speech, which develops naturally, L2 requires explicit teaching. These features make transcription challenging for standard ASR systems.

To address these challenges, an 85-hour corpus with verbatim transcriptions of spontaneous English speech from Swiss primary school students was collected as part of the ChaLL project. By fine-tuning the “wav2vec2-XLSR-300M” model on this targeted corpus, Michot et al. (2024) were able to outperform other ASR systems (including Whisper-Large) in terms of error preservation and surpasses English TTS models of comparable size ( 300M parameters) by a large margin in terms of Word Error Rate (WER). To evaluate systems’ capacity to preserve learners’ errors, a novel metric Word-Based Error Preservation Rate (WEPR) was proposed Michot et al. (2024). The resulting “ChaLL-300M” model shows the necessity of using targeted data, to fine-tune a ASR system, which is required in downstream tasks.

## 1.2 Problem Statement

However, while fine-tuning pre-trained ASR models on targeted datasets has yielded improvements in transcription accuracy and error preservation, limitations remain. The spontaneous nature of children’s speech in second language acquisition introduces considerable variability that the fine-tuned ASR system still struggles to handle. Although the “ChaLL-300M” model has demonstrated the impact of targeted data, it also underscores the continued need for task-specific data and methodologies.

Accordingly, the superordinate goal of this thesis is to take the next step in developing a ASR system in this unique context. Several approaches can enhance ASR performance, such as leveraging alternative models or optimizing hyperparameters. Although training larger, state-of-the-art ASR systems like Whisper offers significant potential compared to acoustic models like XLSR, the emphasis of this work remains on the data. But as the collection and annotation is both labor-intensive and expensive, the focus is on using data augmentation to generate additional data for ASR training. By artificially increasing the size and diversity of the training data, data augmentation can help mitigate the scarcity of annotated speech corpora.

Generating synthetic children-speech data with controlled linguistic content could enhance ASR models’ ability to preserve learner errors in general but also on certain types of errors. If it is possible to replicate real data through data augmentation, this approach could further enhance error-handling capabilities, enabling ChaLL to provide more accurate feedback and foster more effective language learning.

### 1.3 Research Objectives

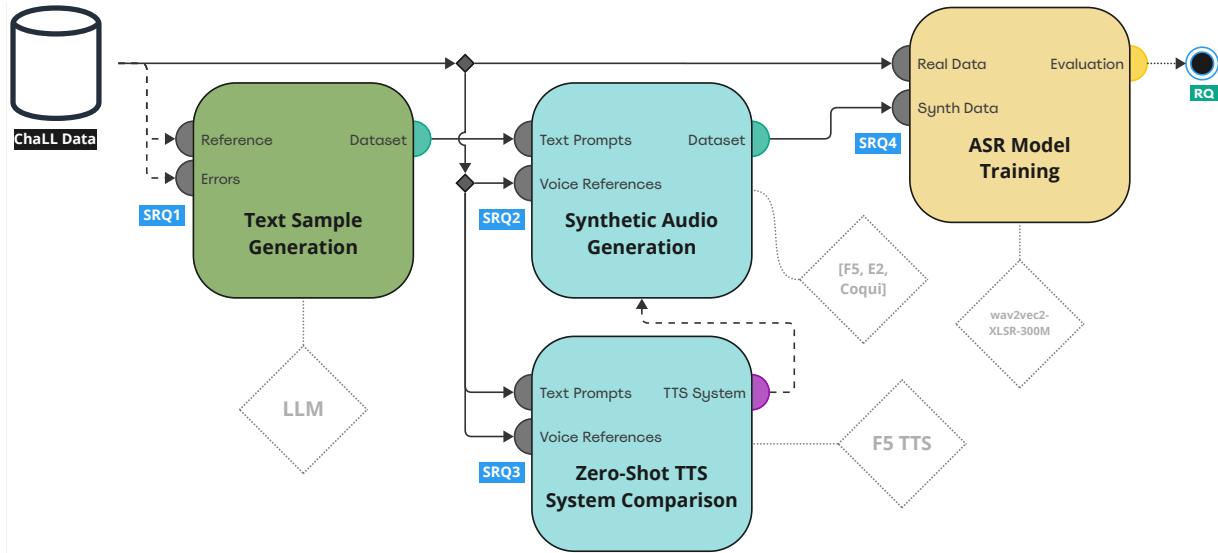
Building on recent advancements in both Text-to-Speech (TTS) and text generation using Large Language Models (LLMs) the central research question of this thesis is formulated as:

**Research Question:** What is the impact of augmenting real training data with synthetic speech on ASR accuracy and error preservation when fine-tuning ASR models for transcribing the speech of second-language learners, particularly children?

This research question is addressed by breaking it down into objectives, each providing a part of the overall answer. The scope of this work is structured around three primary objectives. The first objective is to generate controlled text samples that replicate the spontaneous speech of child second-language learners, with a particular emphasis on incorporating predefined errors (See Section 5). The second objective involves evaluating the most suitable zero-shot TTS system for voice conversion in children's voices and the creation of a synthetic speech corpus (See Section 6). Finally, the third objective focuses on fine-tuning TTS models, particularly "wav2vec2-XLSR-300M", using different proportions of real and synthetic data to assess the impact of data augmentation on error preservation and transcription accuracy (See Section 7). These objectives lead to the formulation of the following sub-research questions:

1. **SRQ1** (See Section 5): How can text samples be generated to accurately replicate the spontaneous speech patterns and learner-specific errors observed in the ChaLL corpus?
2. **SRQ2** (See Section 6): What are the capabilities of existing zero-shot voice-cloning TTS systems in synthesizing children's voices and texts from the ChaLL corpora, and how can their similarity (SMOS) and naturalness (CMOS) be evaluated?
3. **SRQ3** (See Section 6): What methodologies can be employed to efficiently generate a 100-hour synthetic speech corpus by synthesizing generated text samples with children's voice references, while ensuring the quality and authenticity of the TTS data?
4. **SRQ4** (See Section 7): How does the integration of different proportions of synthetic speech data with real data influence the transcription accuracy and error preservation capabilities when finetuning ASR models?

Figure 1.1 illustrates the three main objectives defining the scope of this thesis: text sample generation, audio dataset generation (including TTS system selection), and ASR model training with mixed proportions of real and synthetic data. The ChaLL corpus



**Figure 1.1:** Overview of the three main objectives defining the scope of this thesis and their relationships: text sample generation, audio dataset generation (including TTS system selection), and mixed TTS training. The ChaLL corpus serves as the foundational input for all objectives, providing reference data for text generation, input for TTS evaluation, source voices for synthetic speech creation, and real data for mixed training of ASR models. Each objective contributes to assessing the impact of data augmentation on ASR accuracy and error preservation.

serves as the foundation, providing reference data for text generation, input for TTS system comparison, source voices for synthetic dataset creation, and real data for mixed ASR training.

The first objective involves using LLMs (GPT4o) to generate text samples that replicate the spontaneous speech patterns and learner-specific errors observed in the ChaLL corpus. Next, a systematic evaluation of existing TTS systems is conducted to identify the most suitable system for synthesizing speech. Using the selected system, a synthetic speech corpus is created by pairing the generated text samples with children's voice references. Finally, ASR models are trained with varying proportions of real and synthetic data to assess whether data augmentation improves transcription accuracy and the preservation of learner-specific errors, addressing the central research question.

## 1.4 Key Findings & Contributions

This work presents a multi-step data augmentation pipeline for generating child-like speech with controlled errors. The resulting synthetic dataset was used to train ASR models with varying ratios of real and synthetic data, evaluating its impact on transcription accuracy and error preservation.

The pipeline includes a text sample generator that produces learner-like text samples with controlled grammatical errors using a modular prompt-based approach with GPT-4o. This process generated

34,000 parallel text samples of both correct and incorrect sentences, replicating the errors made by CEFR A1–A2 learners.

A comparative analysis of multiple zero-shot TTS systems identified the best performer based on speaker similarity (SMOS) and naturalness (CMOS). Among the tested systems (F5, E2, Coqui), F5 TTS achieved the highest SMOS and CMOS scores, making it the most suitable for zero-shot TTS with voice cloning in this setting. Using this system, a 100-hour corpus of synthesized child speech was created to support ASR training.

The synthetic speech dataset was then used in mixed-training experiments, combining real and synthetic data to evaluate its effect on ASR performance. The results show that incorporating synthetic speech data improves ASR transcription accuracy and error preservation, but the gains are inconsistent across configurations. The best-performing models achieve modest improvements, with 80 hours of synthetic data reducing WER by 1.19% and improving WEPR by 2.73% compared to real-only training.

## 1.5 Structure of the Thesis

This thesis is structured to first provide a comprehensive background, then detail experimental methodologies and results, and finally discuss findings in the context of the research objectives. Chapter 1 (Introduction), along with Chapter 2 (Fundamentals) and Chapter 3 (Related Work), establishes the foundation of this research. Chapter 2 provides an overview of the core technologies and methodologies, including TTS systems, LLMs, and ASR. In contrast, Chapter 3 reviews prior studies on second-language learners, particularly children, exploring existing approaches for generating erroneous text samples, creating synthetic audio with children’s voices, and training ASR systems with targeted datasets.

The experimental framework is detailed in Chapter 4 (Experiment Setup), which explains the design and implementation of experiments and their relevance to the research objectives. This chapter serves as the basis for the subsequent experimental chapters: Chapter 5 (Text Sample Generation), Chapter 6 (Audio Dataset Generation), and Chapter 7 (ASR Training). Each of these chapters focuses on a specific research objective, detailing the methods, experiments, and results.

Finally, Chapter 8 (Conclusion and Future Work) summarizes the most important findings, discusses their implications in the context of the research questions, highlights the limitations of the work, and outlines potential directions for future research.

# Fundamentals

# 2

This chapter addresses the technologies utilized in this study. Chapter 3, in contrast, explores content-focused research. Large Language Models (LLMs), described in Section 2.1, are used to generate text samples that replicate spontaneous speech patterns and learner-specific errors. Text-to-Speech (TTS) systems, covered in Section 2.2, are evaluated in this thesis to identify the most suitable system for synthesizing speech, enabling the creation of a synthetic speech corpus. Automated Speech Recognition (ASR) models, discussed in Section 2.3, are trained with varying proportions of real and synthetic data to assess whether data augmentation improves transcription accuracy and error preservation. Finally, Grammatical Error Correction (GEC), described in Section 2.4, is used to evaluate the preservation of learner-specific errors.

## 2.1 Large Language Models

Since the introduction of the Turing Test in the 1950s, researchers have pursued the development of machines capable of mastering language intelligence. Language, a sophisticated system of human expression structured by grammatical rules, presents a considerable challenge for AI to comprehend and utilize effectively (W. X. Zhao et al., 2024). Over the past two decades, language modeling has emerged as a central approach to advancing language understanding and generation, transitioning from statistical methods to neural-based models.

A LLM is a deep learning model designed to process and generate human-like text by being trained on extensive datasets of written language. The term LLM specifically refers to Pre-Trained Language Models (PLMs) with significant parameter scales, often containing tens or hundreds of billions of parameters. The most advanced LLMs use the Transformer (Vaswani et al., 2017) framework, which relies on a self-attention mechanism to effectively identify and manage relationships over long sequences of input data. These models excel in a wide range of tasks, including natural language applications such as summarizing text, translating languages, and answering questions. Interestingly, when the parameter scale exceeds a certain level, these enlarged language models not only achieve significant performance improvements in language modeling, they also exhibit special abilities, such as in-context learning, that are absent in smaller-scale models (W. X. Zhao et al., 2024).

Research on LLMs has recently seen significant advancements driven by both academia and industry, with one notable milestone being the launch of ChatGPT<sup>1</sup>. The idea of using LLMs in this

2.1	Large Language Models . . . . .	7
2.2	Text-to-Speech . . . . .	11
2.3	Automated Speech Recognition (ASR) . . . . .	16
2.4	Grammatical Error Correction . . . . .	22

1: <https://openai.com/blog/chatgpt/>

thesis is to generate text samples similar to the real text samples from the ChaLL corpus. By leveraging the in-context learning capabilities of these models, the aim is to create samples with the same spontaneous characteristics and grammatical errors as the real data. The methodology described in Chapter 5 utilizes GPT (Section 2.1.1) to create controlled text samples through the use of prompt chains and refined prompts (Section 2.1.2).

2: Decoder-only LLMs (Brown et al., 2020; Le Scao et al., 2023; Radford et al., 2019; Radford, 2018; Touvron et al., 2023)

**Decoder-only** architectures form the core of many LLMs<sup>2</sup> designed for generative tasks. This design makes decoder-only models particularly well-suited for autoregressive generation, where the model predicts the next token based on a sequence of prior tokens. Unlike encoder-decoder (Vaswani et al., 2017) architectures, that separately encode input and generate output, decoder-only models process input and output through the same network, streamlining the architecture for tasks like open-ended text generation and language modeling.

**Decoding strategies** refers to the methods used to select the next token from a model's predicted probability distribution. The most straightforward approach is *greedy decoding*, where the token with the highest likelihood is selected. *Beam search* (Sutskever, 2014) mitigates the risk of missing high-probability sequences by tracking the most likely hypotheses, at each step and selecting the one with the highest overall probability. While it often outperforms greedy decoding, it does not always guarantee the globally most likely sequence.

Sampling-based methods offer an alternative decoding strategy by selecting the next token randomly, according to the probability distribution, aiming to increase randomness and enhance diversity in generated outputs. Unlike deterministic methods (like greedy decoding or beam search), which focus on optimizing for the most probable sequence, sampling-based techniques select tokens probabilistically based on their predicted likelihoods. Various strategies have been proposed to enhance generation quality by reducing or avoiding the selection of tokens with low probabilities (W. X. Zhao et al., 2024):

- ▶ *Temperature Sampling*. Temperature sampling adjusts the randomness of token selection by modifying the temperature coefficient ( $t$ ) in the softmax function<sup>3</sup>. Lowering  $t$  prioritizes high-probability tokens, making the output more deterministic, while raising  $t$  increases randomness. At  $t = 1$ , the process performs standard random sampling; as  $t \rightarrow 0$ , it approximates greedy search, and as  $t \rightarrow \infty$ , it degenerates to uniform sampling.
- ▶ *Top-k Sampling*. Top-k sampling differs from temperature sampling by limiting token selection to the top  $k$  most probable tokens, ignoring the rest.
- ▶ *Top-p Sampling*. Also known as nucleus sampling, top-p sampling dynamically adjusts the token selection based on

3: The probability of the  $j$ -th token over the vocabulary is computed as:

$$P(x_j|x_{<i}) = \frac{\exp(l_j/t)}{\sum_{j'} \exp(l_{j'}/t)}$$

where  $l_j$  represents the logits of each word, and  $t$  is the temperature coefficient.

context. It samples from the smallest set of tokens whose cumulative probability meets or exceeds a threshold  $p$ . This set is created by sequentially adding tokens, sorted in descending order of probability, until the cumulative value surpasses  $p$ .

### 2.1.1 GPT

OpenAI has achieved two significant milestones, ChatGPT (OpenAI, 2022) and GPT-4 (OpenAI, 2023a), which have substantially advanced the capabilities of existing AI systems. ChatGPT builds on the GPT-3.5 and GPT-4 architectures, optimized specifically for dialogue through a unique dataset combining human-generated conversations and the InstructGPT (Ouyang et al., 2022) dataset. GPT-4 expanded to multimodal inputs and showcased superior performance (Bubeck et al., 2023) in complex tasks and improved safety to mitigate harmful outputs. Building on these advancements, OpenAI introduced GPT-4o (OpenAI, 2023b), a further evolution designed to optimize performance, cost-efficiency, and accessibility.

The OpenAI API supports several decoding strategies, including greedy search (temperature  $t = 0$ ), beam search (best\_of parameter), temperature sampling (temperature), and nucleus sampling (top\_p). Additional parameters like presence\_penalty and frequency\_penalty allow control over repetition in the generated text. Notably, even with identical inputs and hyperparameters, the API may produce varying outputs. Setting temperature to 0 reduces randomness, resulting in more deterministic outputs with minimal variability.

### 2.1.2 Prompting

Prompt creation, also referred to as prompt engineering, involves designing prompts to optimize LLMs for specific tasks. According to W. X. Zhao et al. (2024), this process leverages four key elements:

1. **Task Description:** Clear and specific instructions that guide the LLM on the task objectives. Detailed clarifications and highlighted keywords are useful for tasks with unique input or output formats. Providing examples of desired input-output pairs (few-shot examples) can further enhance the model's understanding of the task. Additionally, structuring prompts with consistent and model-friendly formats, such as specific symbols or delimiters (e.g., , " "), helps improve comprehension and alignment with task requirements.

2. **Input Data:** Describes the data to be processed by the LLM. For standard data, natural language suffices, while structured data (e.g., tables, knowledge graphs) often require transformation into linearized sequences or programming formats for better processing.
3. **Contextual Information:** Background information, such as relevant documents or in-context task examples, helps improve task-specific performance. High-quality and relevant context boosts accuracy, especially for complex tasks.
4. **Prompt Style:** Tailored expression of prompts to align with the specific LLMs strengths. Styles may include prefixes like “Let us think step by step” for logical reasoning or role-setting phrases such as “You are an expert in this domain.”

### 2.1.3 In-Context Learning (ICL)

In-Context Learning (ICL) is a prompting technique introduced with GPT-3 (Brown et al., 2020). Unlike fine-tuning and other optimization techniques, in-context learning does not alter the model’s underlying parameters but relies solely on examples provided as demonstrations (W. X. Zhao et al., 2024). In-context learning allows models to perform tasks by conditioning on examples provided in the input, without requiring fine-tuning or explicit task-specific training. The idea is to show the model what to do, instead of just telling. The design of demonstrations, including their selection, format, and order, significantly impacts the effectiveness of ICL (W. X. Zhao et al., 2024). A typical approach to implementing in-context learning is through one-shot or few-shot learning:

- ▶ **Zero-Shot Learning:** The model performs a task based solely on a natural language instruction without any demonstration.
- ▶ **One-Shot Learning:** The model is provided with a single example of the task to guide its response.
- ▶ **Few-Shot Learning:** The model leverages a small number of task examples to better understand and perform the task.

### 2.1.4 Prompt Chaining

Prompt Chaining divides a task into smaller, sequential prompts, where the output of one step becomes the input for the next. Each prompt focuses on a specific aspect of the task, enabling iterative refinement and improvement. This approach is especially effective for tasks requiring gradual enhancement or involving multiple interconnected steps.

## 2.2 Text-to-Speech

Since speech is the most efficient and natural form of human communication (Pinker, 2003), extensive research has focused on enabling computers to replicate human-like speech. The process of converting text into audible speech is known as speech synthesis or Text-to-Speech (TTS). Recently, TTS systems have seen significant improvements (J. Kim et al., 2020; Ren et al., 2019, 2022), reaching a degree of naturalness that is identical to human speech (Tan et al., 2024). In this thesis, TTS is used to generate synthetic speech data for ASR model training. This is achieved by synthesizing the generated text samples using voice cloning (Section 2.2.2) with real children's voices.

TTS systems are designed to transform any given text into natural and realistic speech. This process involves two main steps: first, text analysis, where the input text is converted into symbolic or phonetic representations used for modeling sound and speech patterns; and second, the generation of speech waveforms to produce the actual audio. Speech synthesis techniques are broadly categorized into two types: traditional machine learning methods and advanced deep learning approaches.

**Traditional machine learning methods** in TTS include concatenative (Khan & Chitode, 2016) and statistical parametric (Zen et al., 2009) approaches. Concatenative synthesis involves assembling recorded speech segments into complete utterances, with specialized techniques ensuring smooth transitions between these segments. In contrast, statistical parametric synthesis utilizes mathematical models like Hidden Markov Models (HMMs) (Tokuda et al., 1995) to generate speech. HMMs extract speech parameters such as excitation (fundamental frequency) and spectral features from the training speech recordings using vocoders (voice encoder) and use them to produce the speech waveform during the synthesis process (Ling et al., 2015). While these more traditional methods have been foundational in TTS development, they often face challenges in producing highly natural and expressive speech.

**Advanced Deep Learning Approaches** address these limitations using Deep Neural Networks (DNNs) (Qian et al., 2014; Zen et al., 2013) and Recurrent Neural Networks (RNNs) (Achanta et al., 2015). These approaches significantly improve the quality of synthesized speech and enhance learning capabilities such as voice conversion (Sisman et al., 2021). Further, Long Short-Term Memory (LSTM) RNNs (LSTM-RNNs) have demonstrated improved performance by effectively modeling sequential dependencies (Zen & Sak, 2015). Each output is influenced not only by the current input but also by the context provided by previous inputs in the sequence. The ability to capture long-term dependencies in sequential data generates more natural and coherent speech, but at the

same time, the recurring architecture of LSTM-based RNNs limits their scalability and training (Kaur & Singh, 2023).

### 2.2.1 State-of-the-Art TTS

Recent deep learning-based speech synthesizers use **Autoregressive (AR) Models** (Oord et al., 2016) to generate high-fidelity waveforms by first converting text into mel-spectrogram and then using vocoder to produce speech. This approach requires only audio data and corresponding transcripts for training (Ping et al., 2018; J. Shen et al., 2018). Models such as WaveNet (Oord et al., 2016) and Tacotron (Y. Wang et al., 2017) produce speech one sample or frame at a time, building on prior outputs. These models effectively produce high-quality, natural-sounding audio by capturing sequential data dependencies. They achieve precise control over audio quality and speech naturalness using architectures such as Convolutional Neural Network (CNN) and attention mechanisms, which align text with speech and accurately capture local audio patterns. But the sequential nature limits scalability (and parallelism), slows inference, and can cause errors like skipped or repeated words. Advanced models like Tacotron-2 and Deep Voice 3 (Ping et al., 2018) mitigate these issues by combining convolutional and recurrent networks.

Unlike AR models, **Non-Autoregressive (NAR) Models** achieve faster inference through parallel processing. A key challenge for NAR zero-shot TTS models is aligning input text with output audio due to their differing lengths, typically requiring either an explicit duration model or a specialized architecture. NaturalSpeech 2 (K. Shen et al., 2023), NaturalSpeech 3 (Ju et al., 2024), and Voicebox (Le et al., 2023) address this with frame-wise phoneme alignment during training. In contrast, Matcha-TTS (Mehta et al., 2024) relies on a duration model during inference (monotonic alignment search). More recent E3 TTS (Gao et al., 2023) introduces cross-attention for alignment, requiring a carefully designed U-Net architecture.

### 2.2.2 Voice Cloning

Advancements in TTS enable the generation of natural speech for any speaker using just a few seconds of audio, known as an audio prompt. Voice cloning, allows synthesizing voices of speakers not included in the model's training data. Zero-shot multi-speaker TTS (Casanova et al., 2021, 2024; Chen et al., 2024; Eskimez et al., 2024; Jia et al., 2018) extracts unique vocal features from brief speech samples to replicate a voice, eliminating the need for extensive recordings or retraining.

### 2.2.3 E2-TTS

E2 TTS (Embarrassingly Easy Text-to-Speech) (Eskimez et al., 2024) is a fully NAR zero-shot TTS system with a simple architecture compromising of two main modules: a flow-matching-based mel spectrogram generator and a vocoder. The spectrogram generator, using a Transformer with U-Net-style skip connections, predicts masked audio regions based on context and aligns them with input text converted into a character sequence with filler tokens. The vocoder then converts the generated mel spectrogram into speech. Unlike more complex models requiring duration estimation or grapheme-to-phoneme conversion, Compared to Voicebox and NaturalSpeech 3, E2 TTS achieves state-of-the-art naturalness, intelligibility, and speaker similarity without the need for duration estimation or grapheme-to-phoneme conversion.

### 2.2.4 F5-TTS

F5-TTS (Fairytaler Fakes Fluent and Faithful Speech with Flow Matching) (Chen et al., 2024) is another fully NAR TTS system trained on 100K hours of multilingual speech. Like E2-TTS, it employs a simple architecture without phoneme alignment, duration predictors, text encoders, or semantically infused codec models. However, F5-TTS addresses E2-TTS's limitations by using ConvNeXt V2 (Woo et al., 2023) modules to refine text representations, improving text-to-speech alignment, robustness, and convergence speed.

Additionally, F5-TTS uses **Sway Sampling** strategy during inference. Sway Sampling enhances inference in flow-matching models by prioritizing critical flow steps. In flow matching, the model learns to transform a simple initial distribution (e.g., Gaussian noise) into the target distribution (speech features) by optimizing a neural network to guide the transformation at each step. Using Sway Sampling, early steps, crucial for structural accuracy (e.g., alignment), are emphasized by biasing the sampling distribution using the formula:

$$f_{\text{sway}}(u; s) = u + s \cdot \left( \cos\left(\frac{\pi}{2}u\right) - 1 + u \right) \quad (2.1)$$

where  $u$  is uniformly sampled, and  $s$  controls the bias:  $s < 0$  favors earlier steps,  $s = 0$  is uniform, and  $s > 0$  emphasizes later steps. This approach improves output naturalness and alignment while optimizing computational efficiency without requiring model retraining.

### 2.2.5 Coqui XTTS

The XTTS (Casanova et al., 2024) system is a multilingual zero-shot NAR TTS model designed to overcome the limitations of existing TTS models that primarily support single or few high-resource languages. XTTS introduces advanced adaptations to the Tortoise (Betker, 2023) model, integrating components like Vector Quantized-Variational AutoEncoder (VQ-VAE), a GPT-2-based encoder, and a HiFi-GAN-based decoder. These components enable robust multilingual training, voice cloning, and faster inference across 16 languages, including low- and medium-resource ones. It achieves state-of-the-art performance in multilingual zero-shot TTS, producing natural, expressive speech and enabling cross-language voice synthesis.

### 2.2.6 TTS Evaluation

To evaluate the generated speech samples, two subjective metrics SMOS and Comparative Mean Opinion Score (CMOS) are used in this work. Additionally, DataSpeech is utilized to tag speech data, enabling an automatic comparison between real and synthetic tagged speech.

### 2.2.7 Similarity Mean Opinion Score (SMOS)

SMOS is a perceptual evaluation metric used to assess the speaker similarity of synthesized speech compared to a reference recording. Evaluators rate how closely the voice in the synthesized audio matches the characteristics of the target speaker's voice, including tone, pitch, and speaking style. The SMOS scale ranges from 1 to 5, where 1 indicates no similarity and 5 represents perfect similarity, with increments of 0.5.

### 2.2.8 Comparative Mean Opinion Score (CMOS)

CMOS is a perceptual evaluation metric used to assess the relative naturalness of synthesized speech when compared to a reference sample. Evaluators rate whether the synthesized speech is "better" (smoother, more natural, and closer to human-like qualities) or "worse" (robotic, unnatural, or distorted) than the reference. The CMOS scale ranges from -3 to +3, where negative values indicate the synthesized speech is worse than the reference, 0 indicates equal quality, and positive values reflect superior quality.

### 2.2.9 DataSpeech

DataSpeech (Lacombe et al., 2024) is a tool developed by Lyth and King (2024) for automatically annotating and preparing speech datasets for TTS model training (e.g., Parler-TTS<sup>4</sup>). By tagging both real and synthetic speech, DataSpeech enables the assessment of how closely synthesized speech matches real speech in terms of speaker attributes, acoustic quality, and linguistic features, complementing subjective metrics.

4: <https://github.com/huggingface/parler-tts>

DataSpeech extracts and appends the following continuous features to the dataset.

- ▶ **utterance\_pitch\_std:** Measures the standard deviation of pitch in the utterance.
- ▶ **utterance\_pitch\_mean:** Measures the average pitch in the utterance.
- ▶ **snr:** The speech-to-noise ratio quantifies the clarity of the audio by assessing noise levels.
- ▶ **c50:** Th reverberation estimates the level of echo or room effect in the audio sample.
- ▶ **speaking\_rate:** Rate of speech measured in phonemes per second.
- ▶ **phonemes:** Used to compute the speaking rate.
- ▶ **pesq:** Wideband Perceptual Estimation of Speech Quality ('P.862.2 : Wideband Extension to Recommendation P.862 for the Assessment of Wideband Telephone Networks and Speech Codecs', 2005). Evaluates speech quality on a scale of -0.5 to 4.5, where higher scores indicate better quality.
- ▶ **si-sdr:** Scale-Invariant Signal-to-Distortion Ratio (Roux et al., 2019) measures intelligibility and provides a proxy for noise estimation.
- ▶ **stoi:** Short-Time Objective Intelligibility (Taal et al., 2010) predicts speech intelligibility in noisy conditions, with scores ranging from 0 to 1.

To enhance interpretability, DataSpeech maps continuous features into predefined keyword bins:

- ▶ **Pitch Bins:** Labels such as "very low pitch," "moderate pitch," or "very high pitch."
- ▶ **Speaking Rate Bins:** Categories like "very slow," "moderate speed," or "very fast."
- ▶ **SNR Bins:** Keywords like "very noisy," "moderate ambient sound," or "very clear."
- ▶ **Reverberation Bins:** Labels like "very roomy sounding" or "very confined sounding."
- ▶ **Speech Quality Bins:** Categories like "bad quality" or "excellent quality."

## 2.3 Automated Speech Recognition (ASR)

This section builds on the technical analysis presented in a previous semester project (Michot, 2024), where foundational aspects of ASR and TTS systems were already explored. The content has been refined and expanded for this thesis to provide a deeper understanding of ASR technologies. As in previous work, the approach involves fine-tuning existing ASR models to assess the impact of using targeted data. In this thesis, the objective is to assess how using synthetic data in addition to real data affects transcription accuracy and error preservation.

Driven by advancements in computational capabilities and access to extensive datasets, ASR or TTS has emerged as a predominant mode of human interaction with various devices (Yu & Deng, 2015). Traditional ASR systems consist of four key components: signal processing and feature extraction, acoustic model, language model, and hypothesis search. Signal processing converts audio signals into feature vectors, the acoustic model scores these based on acoustics and phonetics, the language model estimates word sequence probabilities, and hypothesis search combines AM and LM scores to produce the recognized text (Yu & Deng, 2015). While older ASR systems relied on techniques like Gaussian mixture models and hidden Markov models for acoustic modeling, recent advances favor deep learning methods. This includes methods like the deep neural network-hidden Markov model (Dahl et al., 2011; Hinton et al., 2012) approach, which uses neural networks for improved feature extraction and discrimination. Additionally, End-to-End (E2E) models have gained popularity, combining all ASR components into a single neural network for better accuracy, especially in common languages (Li, 2021). Since most language processing tasks exhibit strong correlations, large pre-trained models based on Self-Supervised Learning (SSL) have emerged.

**End-to-End (E2Es)** models refer to a type of model training where the entire process, from input to output, is handled by a single system without any specific engineering of intermediate steps. E2E models use a single objective function aligned with the ASR objectives to optimize the entire network. Unlike traditional hybrid models that optimize individual components, E2E models thus ensure global optimization (Li, 2021). In addition, E2E models directly generate characters or words in the context of ASR, which simplifies the ASR pipeline compared to complicated manual design steps (acoustic features, pronunciation units, etc.) and extensive ASR expertise required in hybrid approaches (Li, 2021).

E2E models are generally composed of an encoder module, denoted as  $H(X)$ , which transforms an input acoustic frame sequence of length  $T'$  into a higher-level representation  $H(X) = (h_1, \dots, h_T)$ , where  $T$  is usually equal to or less than  $T'$ . Using pairs of utterances and corresponding transcripts as training examples, the goal is to

estimate the conditional distribution over all possible transcripts for given input acoustics:

$$P(C|X) = P(C|H(X)). \quad (2.2)$$

The challenge in estimating  $P(C|X)$  with E2E models is to account for the unknown alignments between the acoustic frame sequences of length  $T$  and the corresponding label sequences of length  $L$  (Prabhavalkar et al., 2023).

Explicit E2E modeling approaches use a latent variable to model alignments, which is then marginalized out during both training and inference (Prabhavalkar et al., 2023). While these approaches differ in the way they define alignments, they share the characteristic of introducing a blank symbol and defining an output probability distribution over the symbols (Prabhavalkar et al., 2023). CTC-based models enumerate explicit alignments before aggregating them for soft alignments, assuming label independence during the enumeration (D. Wang et al., 2019). In contrast, RNN-transducer models also enumerate explicit alignments but do not make assumptions about label independence, leading to differences in path definition and probability calculation (D. Wang et al., 2019). Unlike explicit alignments models, attention-based encoder-decoder models skip the enumeration of hard alignments altogether and directly calculate soft alignments using the Attention mechanism (D. Wang et al., 2019).

**Connectionist Temporal Classification (CTCs)**, as proposed by D. Wang et al. (2019), is a technique allowing the integrating of deep neural networks into ASR. CTC addresses the challenge of mapping variable length speech input sequences to output sequences by serving as a loss function (Li, 2021). Unlike traditional methods, CTC eliminates the need for explicit alignment during loss calculation, making it alignment-free and advantageous for an E2E approach in speech recognition. This allows for the direct output of target transcriptions without requiring intermediate units. To handle the length of the output labels compared to the input sequence in the ASR, a blank label was introduced, and the repetition of labels was allowed (Li, 2021). This innovation enables the creation of CTC paths aligned with the length of the input sequence.

In the initial path probability calculation, the encoder transforms the input sequence of length  $T$  into a feature sequence of the same length, which is then processed to generate a probability distribution sequence (D. Wang et al., 2019). Each element in this sequence represents the probability of a specific label or the blank label at each time step, explicitly aligning each input frame to a corresponding label. Path aggregation then reduces the length of the output paths by merging identical labels and removing blank labels, resulting in the final label sequence. While the assignment

of the input to the sequence is explicitly aligned, it is soft aligned in the aggregation step without the need for explicit alignment according to a specific path (D. Wang et al., 2019).

The CTC algorithm assumes label independence in the output sequence, which simplifies speech modeling as the encoder can focus on acoustics rather than complex speech modeling (D. Wang et al., 2019). However, this conditional independence assumption in CTC is a point of criticism (Li, 2021). To address this criticism, the attention mechanism (Das et al., 2018; Salazar et al., 2019) is used to introduce implicit language modeling across speech frames. Attention-based CTC models tackle the independence assumption by enhancing the encoder without altering the CTC objective function, while maintaining the simplicity of CTC (Li, 2021). Recent studies boost CTC by replacing the underlying LSTM with Transformer (Vaswani et al., 2017), enabling a more powerful attention mechanism (Miao et al., 2019). Additionally, self-supervised learning technologies (Baevski et al., 2020; Hsu et al., 2021) further enhance CTC by learning a robust semantic representation.

**Attention-based Encoder-Decoder (AED).** The incorporation of a decoder results in a sequence-to-sequence model, framing speech recognition as a language generation task that utilizes input speech to generate corresponding label sequences (D. Wang et al., 2019). The AED model is an E2E model with an encoder network, an attention module, and a decoder network. Its training objective is to minimize a calculated probability, aiming to improve alignment between speech signals and label sequences (Li, 2021). The encoder network transforms input feature sequences into high-level hidden feature sequences, while the attention module computes attention weights between the previous decoder output and the encoder output of each frame. The decoder network generates its output in an autoregressive manner based on the previous label outputs and a context vector (Li, 2021). To address alignment issues between the speech signal and label sequence, AED models are often optimized with a CTC model in a multi-task learning framework by sharing the encoder (Li, 2021). This training strategy, widely adopted in most AED models (S. Kim et al., 2017; Ueno et al., 2018) improves convergence, and mitigates alignment issues (Li, 2021).

### 2.3.1 wav2vec 2.0

Models like wav2vec 2.0 (Baevski et al., 2020) and HuBERT (Hsu et al., 2021) have demonstrated the simplicity and effectiveness of combining CTC with transformer encoder models. One key feature of wav2vec 2.0 is its efficient semi-supervised training. The model is first pre-trained using masked language modelling on an extensive dataset of unlabeled speech, followed by fine-tuning on a smaller labeled dataset. Remarkably, the authors demonstrated that fine-tuning on just one hour of labeled speech data could

outperform previous SOTA systems trained on significantly larger labeled datasets.

The architecture comprises a feature encoder, context network, quantization module, and contrastive loss. The latent feature encoder  $f : X \rightarrow Z$  reduces the dimensionality with convolutional layers that process raw waveforms  $X$  into latent representations  $z_1, \dots, z_T$ . Each  $z_t$  corresponds to around 25 milliseconds of audio. These representations are then fed to the context network, which follows a Transformer architecture  $g : Z \rightarrow C$ , resulting in context representations  $c_1, \dots, c_T$ . Given the context representations, self-attention is used to capture dependencies across the entire sequence of latent representations end-to-end (Baevski et al., 2020)

Transformers encounter challenges in processing speech due to the continuous nature of spoken language. Phones could serve as a discrete system, but prior labeling of the entire dataset would be required, preventing pre-training on unlabeled data (Boigne, 2021). Thus, wav2vec 2.0 discretizes the output of the feature encoder  $z_t$  to a finite set of speech representations via product quantization (Boigne, 2021). They suggest learning discrete speech units by sampling from the Gumbel-Softmax (Jang et al., 2017) distribution. The discrete speech units consist of concatenated codewords  $V$  sampled from  $G$  codebooks (groups). Wav2vec employs 2 groups, each with 320 possible words, resulting in a maximum of 102,400 theoretical speech units. Finally, this results in a quantized representation  $q_t$  for a latent speech representation  $z_t$ .

The pre-training phase involves unlabeled speech data and a contrastive task with a mask applied in the latent space (Baevski et al., 2020). Contrastive learning involves comparing different samples and grouping them into similar or dissimilar clusters in the latent space (Le-Khac et al., 2020). The model utilizes this mechanism in pre-training to predict the correct quantized representation  $q_t$  using the Transformer-generated contextualized representation  $c_t$ . About half of the latent feature vectors undergo random masking in the latent space, are then predicted by the Transformer, and ultimately compared to the quantized representation using the contrastive loss (Baevski et al., 2020). The final loss  $L$  is calculated as the sum of the contrastive loss and an additional diversity loss.

While the contrastive loss optimizes the transformer, the diversity loss promotes equal utilization of all codewords (Baevski et al., 2020). When fine-tuning, quantization is not applied, and instead, a randomly initialized linear projection layer is added onto the context network in  $C$  classes representing the vocabulary. Fine-tuning involves standard CTC loss and a modified version of SpecAugment (Radford et al., 2018) masking as a regularization technique (Baevski et al., 2020).

**Wav2Vec 2.0 XLSR**, proposed by Babu et al. (2021), is a large-scale, cross-lingual speech representation model based on the wav2vec 2.0 architecture by Baevski et al. (2020). Inspired by Wav2Vec2-XLSR-53 (Conneau et al., 2020), Babu et al. (2021) aimed to enhance diversity by using a larger amount of unlabeled training data, including 436k hours from various sources, with VoxPopuli corpus (C. Wang et al., 2021) being the most significant. The dataset encompasses 128 languages, a notable increase from the 53 in XLSR53. The model comes in three versions based on training parameters: 300 million, 1 billion, and 2 billion mode (Babu et al., 2021).

### 2.3.2 Whisper

Whisper (Radford et al., 2022) is an ASR system, developed using a large, diverse dataset of 680'000 hours of multilingual audio from the web, making it robust against various accents, background noise, and technical language. The system architecture consists of an encoder-decoder Transformer, processing audio in 30-second segments and converting them into log-Mel spectrograms for further processing. Unlike other models that use smaller, closely paired datasets or unsupervised audio pretraining, Whisper excels in zero-shot performance across varied datasets, despite not being the top performer in specialized benchmarks like LibriSpeech. Particularly notable is its proficiency in TTS translation, outperforming supervised SOTA models in tasks like CoVoST2 to English translation without specific fine-tuning (Radford et al., 2022).

### 2.3.3 Evaluation Metrics for ASR

This chapter provides an overview of metrics used to evaluate ASR. These are necessary for systems evaluation and for developing an error-preservation measurement strategy.

**Bilingual Evaluation Understudy (BLEU)**. While BLEU (Papineni et al., 2002) is more commonly associated with machine translation, it can also be adapted for ASR tasks. It measures the overlap of n-grams between the hypothesis and reference and can be used for evaluating the fluency of generated text. BLEU Scores range between 0 and 1.

**Word Error Rate (WER)** metric calculates the percentage of incorrect words in the hypothesis compared to the reference. A lower WER in speech-to-text means better accuracy in recognizing speech. WER can also be applied without considering letter case when case sensitivity is not essential. The WER can be calculated as:

$$\text{WER} = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C} \quad (2.3)$$

where  $S$  is the number of word substitutions,  $D$  is the number of word deletions,  $I$  is the number of word insertions,  $C$  is the number of correct words, and  $N$  is the total number of words in the reference.

**Character Error Rate (CER)** Compared to WER, the CER offers a more detailed analysis by calculating the proportion of incorrect characters in the hypothesis compared to the reference:

$$\text{CER} = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C} \quad (2.4)$$

where  $S$  is the number of character substitutions,  $D$  is the number of character deletions,  $I$  is the number of character insertions,  $C$  is the number of correct characters, and  $N$  is the total number of characters in the reference.

**Character Level F-Score (chrF)** (Popović, 2015) focuses on character n-grams instead of word n-grams like BLEU. chrF is language-independent and has shown promising results in correlating with human evaluations. chrF is less sensitive to sentence tokenization and provides partial reward for incorrectly spelled words. Is calculated as:

$$\text{chrF}(\beta) = (1 + \beta^2) \cdot \frac{\text{CHRP} \cdot \text{CHRR}}{\beta^2 \cdot \text{CHRP} + \text{CHRR}} \quad (2.5)$$

where  $\text{CHRP}$  is the percentage of matching n-grams in the hypothesis,  $\text{CHRR}$  is the percentage of reference character n-grams present in the hypothesis, and  $\beta$  is a parameter for tradeoff.

This algorithm compares sequences, such as a reference and one or more hypotheses, and identifies matches, substitutions ( $S$ ), insertions ( $I$ ), and deletions ( $D$ ) at the word level. WEPR focuses exclusively on word pairs where the reference word is marked with an error annotation.

**Word-Based Error Preservation Rate (WEPR)** (Michot et al., 2024) measures the ASR system's ability to retain errors made by speakers. It is calculated using a custom phonetic word-level alignment algorithm, which utilizes annotated error information in the reference transcripts. This algorithm aligns two or more sequences and identifies matches, substitutions ( $S$ ), insertions ( $I$ ), and deletions ( $D$ ) at the word level. WEPR focuses exclusively on word pairs where the reference word is marked with an error annotation and is defined as:

$$\text{WEPR}(A) = \frac{S + D}{N} \quad (2.6)$$

where  $A$  is the set of annotations that are considered (e.g., {@!, @g}),  $S$  and  $D$  are the number of substitutions and deletions, respectively,

where the reference word contains an error annotation and  $N$  is the total number of reference words that contain an error annotation.

This metric evaluates the proportion of annotated errors preserved in the ASR output, highlighting its fidelity in capturing and retaining speaker errors.

## 2.4 Grammatical Error Correction

Grammatical Error Correction (GEC) is a technique used to identify and correct grammatical, syntactic, and lexical errors in text. In this work, it is applied to correct transcripts, enabling a systematic comparison between the original and corrected versions. By utilizing GEC, errors in transcripts can be detected and analyzed with the ERRANT toolkit (Section 2.4.1). Unlike WEPR, which evaluates overall error preservation, this approach provides a more targeted evaluation of specific error types, offering deeper insights into the strengths of ASR systems in preserving language learners' errors.

### 2.4.1 ERRANT

The ERRANT (Error-Annotated Revision Annotator) framework (Bryant et al., 2017) is a tool designed to analyze grammatical errors in text by providing a standardized method for categorizing and evaluating errors in GEC tasks. It aligns original and corrected sentences, identifies edits, and assigns each edit to specific error categories, such as spelling, verb tense, or word order, using linguistic rules and language-specific resources to ensure accuracy. ERRANT's methodology involves extracting edits through linguistically informed alignment and classifying them using a rule-based framework (Felice et al., 2016) that relies on universal features, such as lemmas and parts of speech, ensuring broad applicability across languages and datasets.

**Edit Extraction** in the ERRANT tool involves identifying the boundaries of edits between original and corrected sentence pairs. Early efforts, used Levenshtein distance to align tokens and combined adjacent nonmatches to handle multi-token edits. Later, the alignment was enhanced by training a classifier to decide whether edits should be merged. Felice et al. (2016) introduced a linguistically-enhanced alignment algorithm that integrated features like Part-of-Speech (POS) and lemma, achieving superior accuracy in mimicking human edits, which is the method adopted in ERRANT.

**Table 2.1:** A sample alignment between an original and corrected sentence (Felice et al., 2016).

We	took	a	guide	tour	on	center	city	.
We	took	a	guided	tour	of	the	city	center

**Automatic Error Typing** assigns error types to extracted edits. ERRANT employs a rule-based framework inspired by the observation that most error types align with POS categories. About 50 rules were created to classify edits based on POS tags and properties like token insertion, deletion, or substitution, while handling special cases like spelling and word order errors. Specific rules were crafted to distinguish nuanced verb errors, making the system dataset-agnostic. The rule-based classifier is dataset-independent, requires no labeled training data, offers transparent error categorization, and ensures consistency.

The ERRANT framework categorizes errors into 25 main types, which can be prefixed with “M:” (Missing), “R:” (Replacement), or “U:” (Unnecessary) for granular evaluation. This results in a total of 55 possible error types. The complete list of all combinations is in Appendix A.

# 3

## Related Work

3.1 Children’s Speech Corpora . . . . .	24
3.2 ASR for children’s speech and language learners . . . . .	25
3.3 Data Augmentation for Children’s Speech and Language Learners . . . . .	26
3.4 Data Augmentation for Erroneous Texts . . . . .	28

This chapter provides an overview of related research aimed at improving ASR systems for young L2 learners. The discussion begins in Section 3.1, which emphasizes the limited availability of high-quality children’s speech corpora. Section 3.2 examines the role of ASR models tailored to children’s speech and highlights the significance of targeted training data in enhancing system performance. In Section 3.3, the potential of data augmentation techniques to further optimize ASR performance for children is analyzed. Finally, Section 3.4 investigates the application of LLMs in generating synthetic speech corpora containing intentional errors, which can be leveraged to improve ASRs robustness.

This chapter builds on prior work (Michot, 2024; Michot et al., 2024), incorporating overlapping discussions in Sections 3.1 and 3.2 to ensure consistency and provide a coherent context for the advancements explored.

### 3.1 Children’s Speech Corpora.

The MyST Children’s Speech Corpus (Pradhan et al., 2016; Ward et al., 2019) includes 499 hours of conversational speech, with 233 hours manually transcribed, designed for a virtual science tutor aimed at young native English speakers. The OGI Kids’ Speech Corpus (Shobaki et al., 2000) features speech from 1,100 American children in kindergarten through grade 10, primarily scripted words and utterances, with a smaller portion of spontaneous speech. The AusKidTalk corpus (Ahmed et al., 2021) captures speech from Australian children aged 3 to 12, including single words, utterances, and narratives. Additionally, smaller datasets focus on specific tasks, such as read-aloud activities (Eskenazi, 1996) or broader analysis of English children’s speech (Hagen et al., 2003; Lee et al., 1999b). For German, the KidsTalk corpus (Rumberg et al., 2022) provides 25 hours of transcribed continuous speech from children aged 3 to 11. Notably, these corpora are designed for native speaker contexts.

There are limited datasets of children’s speech designed for language learners. The TLT-school collection (Gretter et al., 2020) assesses the English and German proficiency of 9- to 16-year-old Italian native speakers. It includes recordings from about 3,000 students, totaling 275 hours of English and 265 hours of German speech, of which only 16 hours of English and 8 hours of German have been transcribed. Batliner et al. (2005) introduced a 60-hour corpus of speech from children aged 4 to 13, featuring multiple

languages such as English, German, and Swedish, along with English spoken by German, Italian, and Swedish native speakers. The CALL corpus (Baur et al., 2019) contains 38,000 English utterances from Swiss German second- and third-year students, focusing on annotating utterance correctness. These recordings, collected through an online dialogue system prompting speech, include subsets of 6,000 annotated utterances released for shared tasks, emphasizing structured interactions and correctness evaluation.

In contrast, the ChaLL corpus<sup>1</sup> (Michot et al., 2024) focuses on spontaneous speech with transcriptions to train an ASR system which can automatically transcribe learners'. It includes 85 hours of audio recordings from 337 Swiss primary school students (ages 9–14, grades 4–6) learning English. The dataset contains 45,004 utterances segmented from 1,016 recordings, each paired with verbatim transcripts that annotate lexical, grammatical, and pronunciation errors, as well as instances of German word usage (code-switching). Speech was elicited through interactive activities like role-playing and guessing games, designed to encourage authentic communication. Metadata includes school area, grade, recording details, and background noise indicators. Transcriptions include lexical, grammatical, and pronunciation error annotations following detailed guidelines. After filtering, the dataset features 485,770 tokens (10,203 unique types) and 14,396 error-annotated tokens, making it a valuable resource for training ASR systems on the speech of child language learners.

1: The ChaLL corpus is available at <https://huggingface.co/datasets/mict-zhaw/chall>. Due to sensitive data involving minors, access is restricted and requires a collaboration agreement with the original project partners.

## 3.2 ASR for children's speech and language learners.

There is limited research on ASR models designed specifically for children's speech, particularly for non-native language learners. Lu et al. (2022) explored the fine-tuning of wav2vec 2.0 (Baevski et al., 2020) using both native children's speech datasets (MyST and OGI) and non-native speech (TLT), comparing these results to models fine-tuned solely on adult data. Their findings demonstrated that ASR models trained on children's speech outperformed those trained exclusively on adult speech, even for non-native speakers. Similarly, Shivakumar and Narayanan (2022) examined the benefits of fine-tuning ASR models with children's speech data, reaching a similar conclusion: incorporating children's data enhances performance. However, they noted that ASR models trained on adults consistently perform better on adult speech than models trained specifically on children's data perform on children's speech.

Jain et al. (2023) examined the fine-tuning of Whisper models using children's speech datasets (e.g., MyST and PF-STAR), finding significant performance improvements over non-fine-tuned versions. While fine-tuned wav2vec 2.0 models performed slightly better on datasets with similar characteristics to the fine-tuning

data (e.g., MyST or datasets with similar age groups and recording conditions), Whisper demonstrated superior generalization to unseen data, highlighting the trade-off between domain-specific fine-tuning and model robustness. By also fine-tuning OpenAI’s Whisper model on children’s speech datasets (e.g., MyST) and augmenting training data with classroom noise, Southwell et al. (2024) achieved significant reductions in ChaLL, with a 38% relative improvement on MyST (9.2% WER) and a 7% improvement on a noisy classroom speech dataset (ISAT) (54% WER). Additionally, they introduced a novel speed-aware beam rescoring method, leveraging prior knowledge of human speaking rates and large language models to refine hypotheses, further enhancing accuracy.

Michot et al. (2024) focused on accurately transcribing children’s L2 speech while preserving linguistic errors in ASR systems for young, non-native English learners. A model, “ChaLL-300M”, was fine-tuned using the 85 hours of ChaLL corpora, and the Word-Based Error Preservation Rate (WEPR) was introduced to measure error preservation. The model outperformed state-of-the-art systems like “Whisper-Large” and “XLSR” in preserving speaker errors, demonstrating the value of targeted data for enhancing ASR performance in language learning.

### 3.3 Data Augmentation for Children’s Speech and Language Learners

State-of-the-art ASR systems struggle to transcribe children’s speech accurately due to limited high-quality child-specific data. While recording new child speech is ideal, it is often impractical. To address this, data augmentation has emerged as a promising alternative for improving performance in children’s speech recognition tasks. Conventional augmentation techniques, such as speed perturbation, spectral augmentation, pitch shifting, and vocal tract length perturbation, modify specific features of existing speech but do not create new speech. Recent approaches, however, leverage voice conversion techniques to transform source speech into perceptually similar child-like speech, while retaining linguistic content.

Shahnawazuddin et al. (2020) introduced a GAN-based **Adult-to-Child Voice Conversion** framework to transform adult speech into child-like speech for ASR training. This method adjusts the acoustic attributes of adult speech to mimic children’s speech, addressing acoustic mismatches between the groups. Combining converted adult speech with original adult data improved recognition rates for children’s speech, and further reductions in WER were achieved by adding a small amount of actual child speech to training. Similarly, S. Zhao et al. (2023) proposed a voice conversion method using the WORLD vocoder<sup>2</sup> to generate childlike speech for ASR

2: <https://github.com/mmorise/World>

training. This approach modifies adult speech by adjusting fundamental frequencies, formant structures, and vowel lengths based on children’s acoustic characteristics. Speech enhancement was incorporated to reduce noise and improve the quality of converted speech. The method outperformed traditional techniques like spectral warping and vocal tract length normalization, significantly improving ASR performance for children’s speech. Additionally, they also combined synthetic childlike data with a small amount of real child speech to further enhance recognition accuracy.

Zhang et al. (2024) propose **Child-to-Child Conversion** to generate augmented child-like speech. Their method uses monolingual and cross-lingual voice conversion to transform source speech into child-like speech in the same or different languages. Cross-lingual voice conversion, leveraging linguistic diversity, significantly improves ASR performance. Augmenting datasets with child-to-child generated speech via voice conversion significantly reduced WERs, particularly with two-fold augmentation during fine-tuning ASR models.

Unlike voice conversion, which transforms existing speech into forms resembling a target speaker, **TTS with Voice Cloning** generates entirely synthetic speech from text prompts and a minimal recording of the target speaker. However, when applied to children’s speech, the inconsistency of TTS-generated audio—caused by children’s non-standard pronunciations—can negatively impact ASR performance. W. Wang et al. (2021) address this by proposing effective data selection strategies, including speaker embedding similarity and character error rate filtering, to improve TTS data quality. Their i-vector similarity-based selection method achieved up to 14.7% relative CER reduction, emphasizing the importance of targeted TTS data selection. The data was generated using FastSpeech 2. To overcome inconsistency of TTS-generated children’s speech, Jain et al. (2022) propose a method for synthesizing child-like speech by adapting a pre-trained adult speech model with 19 hours of cleaned child speech data. The system produces synthetic child speech with a mean opinion score of 3.95 for intelligibility and 3.89 for naturalness, closely resembling real child speech.

Using TTS systems with voice cloning, new speech can be generated from a reference child voice and arbitrary text prompts. This enables a **Two-Step Data Augmentation** process, where text generation and audio synthesis are handled separately. Cornell et al. (2024) proposes a pipeline for generating synthetic multi-speaker conversational speech data to address the scarcity of in-domain training data for conversational ASR systems. It combines LLMs to generate conversational transcripts and a conversational multi-speaker TTS model, to synthesize speech. The generated data is used to fine-tune Whisper on conversational speech tasks. A similar pipeline approach is proposed by Su et al. (2023).

Most existing approaches address challenges related to either children’s speech or second-language learners’ speech, but this work tackles both simultaneously. Further, while prior methods predominantly focus on improving ASR accuracy in terms of WER, this study emphasizes both accuracy and the preservation of learner-specific errors. To address these challenges, data augmentation must accurately capture the unique characteristics of children’s L2 speech and their common linguistic mistakes. Consequently, while current methods provide a foundation, this work introduces a novel data augmentation approach tailored to these needs. To control both the linguistic content and the children-like voices when generating speech, a two-step data augmentation pipeline emerges as the most promising approach for this work. First, a LLM generates text with controlled linguistic errors reflective of children’s L2 language. Next, a zero-shot TTS with voice cloning synthesizes corresponding multi-speaker speech data.

### 3.4 Data Augmentation for Erroneous Texts

Using a LLM allows for the generation of controllable linguistic content, enabling text that reflects specific linguistic errors. This approach aligns with research related to GEC. The ChatLang-8 framework (Park et al., 2024) leverages LLMs, to generate synthetic data for GEC. By employing a systematic process involving, it creates 1 million sentence pairs with diverse subject types and human-like grammatical mistakes (classified by ERRANT error types). Experimental results demonstrate that ChatLang-8 outperforms traditional datasets in improving model performance for GEC tasks.

Similarly, Potter and Yuan (2024) tackle the complexities of code-switched texts in grammatical error correction. Their work addresses the scarcity of high-quality code-switched training data by generating synthetic sentences and injecting errors using a two-step approach. This method results in one of the first substantial code-switched datasets, enabling significant performance improvements for error correction models, particularly for learners of English as a second language. Both studies underscore the potential of LLMs in generating linguistically rich, error-specific datasets. The ChatLang-8 framework particularly aligns with this work’s objectives.

This work focuses on creating synthetic datasets that reflect grammatical errors made by learners of English as a second language. By combining LLM-based text generation with TTSs voice cloning, this work introduces a novel two-step data augmentation pipeline tailored for language learning applications. Ultimately, while previous work focuses on either children’s speech or grammatical error correction, this study integrates both to improve ASR accuracy and error preservation for L2 learners.

# Pipeline-Based Experiment Setup

This chapter provides an overview of the experiments conducted in this thesis, highlighting their purpose and relations. While methodologies, findings, and conclusions are detailed in later chapters, this chapter focuses on the inputs, outputs, and the flow of data through the proposed pipeline framework described in Figure 4.1. The framework subdivides objectives into modular tasks, enabling flexibility, supporting operation chaining, and allowing service reuse across workflows to reduce redundancy and increase efficiency. This chapter primarily introduces the proposed framework architecture in Section 4.1, followed by a detailed explanation of task and service definitions in Section 4.2, and task execution and tracking in Section 4.3. While this chapter does not present direct results, it is important for the overall understanding.

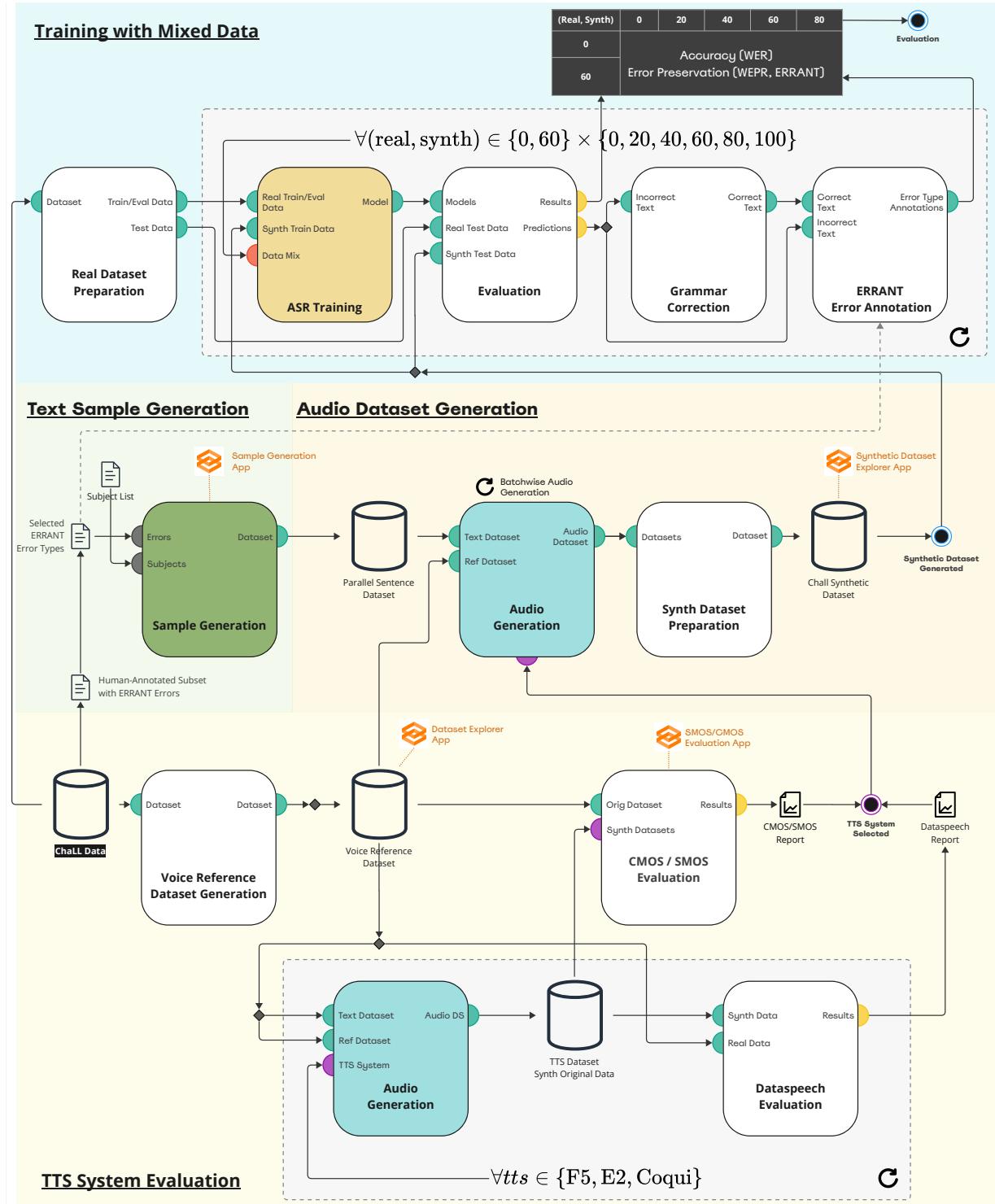
## 4.1 Pipeline Framework Overview

Figure 4.1 illustrates the three main objectives of this work: Text Sample Generation, Audio Dataset Generation, and ASR Training. It is a more detailed version of Figure 1.1 in the Introduction. Each objective consists of tasks requiring specific inputs and configurations to produce outputs. These outputs, or artifacts, are passed between tasks, as depicted by transitions in the figure. Intermediate artifacts, such as datasets, are also shown. Key milestones are highlighted, including *Text Samples Dataset Generated*, *TTS System Selected*, *TTS Audio Dataset Generated*, and *ASR Evaluation*.

The ChaLL corpus (Section 4.1.1) and its ERRANT-annotated (by hand) subset (Section 4.1.2) are used as primary data sources for the pipeline. The ChaLL data functions as a reference for text sample generation, an input for TTS system selection, a voice reference for creating the TTS dataset, and a provider of real data for ASR training. Other datasets are produced as intermediate artifacts. These include a dataset of erroneous text samples, a filtered subset of the ChaLL dataset used as voice references for voice cloning, and the synthesized audio dataset.

The text sample generation objective (Section 5) focuses on generating pairs of grammatically correct and incorrect sentences by leveraging ERRANT error definitions and subject guidelines. These definitions align with the most frequent errors found in the UZH-annotated subset of the ChaLL corpus. The task ultimately produces a dataset of text pairs as its output.

4.1	Pipeline Framework Overview	29
4.2	Modular Service-Based Framework	32
4.3	Experiment Execution & Tracking	34



**Figure 4.1:** The diagram illustrates the individual tasks and processes involved in this work. Each task features distinct inputs and produces specific outputs, with transitions visualizing their relationships. Intermediate results, such as datasets or evaluations, are represented with icons. Gradio apps, highlighted with orange icons and titles, were developed to support interactive evaluations, dataset exploration, and sample generation.

The audio dataset objective involves selecting a suitable TTS system and generating a synthetic dataset using the generated text samples as prompts. Three TTS systems,  $\{F5, E2, Coqui\}$ , are evaluated

using subjective metrics (SMOS, CMOS) and automated analyses (DataSpeech). The process begins by selecting voice references from the ChaLL corpus, resulting in a curated dataset of filtered samples for speech synthesis. The TTS system with the best performance in both subjective and automated evaluations is then used to generate the synthetic audio dataset.

Finally, ASR training assesses the impact of incorporating generated data into model training for children in second language acquisition. This step involves training the wav2vec2-XLSR-300M model with various partitions of real and synthetic data:

$$\forall(\text{real}, \text{synth}) \in \{0, 60\} \times \{0, 20, 40, 60, 80, 100\} \quad (4.1)$$

The result is a comprehensive grid of all real-synthetic data combinations, evaluated using metrics for accuracy and error preservation. Accuracy is assessed with standard metrics such as WER, while error preservation is measured using WEPR and a novel method for evaluating specific error types. This approach involves correcting transcripts with GEC and annotating them using the ERRANT tool, enabling the classification of edits. This process identifies learner errors in the transcripts, which are then compared to the predefined errors used for synthetic data generation.

### 4.1.1 ChaLL Data

To implement data augmentation for replicating erroneous text samples of spontaneous speech by young language learners, the starting point is the data from the ChaLL corpus. In there, speech samples are transcribed verbatim, preserving details such as repetitions, filled pauses, and contractions, while non-English words remain untranslated. Additionally, the transcripts are annotated according to specific conventions, including:

- ▶ @g: (Swiss-)German words
- ▶ @?: Best guesses
- ▶ @!: Errors or missing words

Annotations apply at both word and sentence levels, depending on spacing before the symbol. However, this work focuses on word-level annotations, as sentence-level annotations lack clear assignments. While simple annotations, especially @!, can be used to assess systems overall performance in error preservation (Michot et al., 2024), they fail to distinguish how well the system handles different types of errors.

#### 4.1.2 Human-Annotated Subset with ERRANT Errors

1: <https://prodi.gy/>

As part of the ChaLL error detection module, portions of the annotated ChaLL data were further categorized by the University of Zurich (UZH), a project member of the ChaLL project, using ERRANT error types (Section 2.4.1). This dataset was refined through human annotation of learner grammatical errors, focusing on enhancing the pre-annotated ChaLL @-annotations. Using the Prodigy tool<sup>1</sup>, annotators reviewed and corrected inaccuracies while adding missing details. This dataset remains project-internal and has not yet been publicly released.

The annotation process involved reviewing each sentence to refine error tags. Sentences were first analyzed to determine their intended meaning, and those deemed meaningless or overly erroneous were discarded. Pre-annotated @ errors from transcription agencies and automated systems were then reassessed and adjusted. Error type tags (e.g., Missing, Replacement, Unnecessary) and word category tags were corrected, added, or refined as needed. Untagged errors were labeled appropriately, incorrect tags were fixed, and placeholders or adjacent-word tags were assigned to handle missing words.

Unlike in the ChaLL dataset annotations, hesitations from spontaneous speech (e.g., “uhm”) are not treated as errors. Semantic errors (e.g., “my loved item” instead of “my favourite item”) are annotated with an “R” tag along with the word category of the error (e.g., ADJ in this case). Semantic errors are handled in the same way as lexical errors.

The annotated data was subsequently used in the ChaLL project to train an error detection module. While the annotated samples were available for reference, access to the trained tool itself was not provided for this work. As a result, the tool could not be used to identify errors in either the generated samples or the final transcript evaluation.

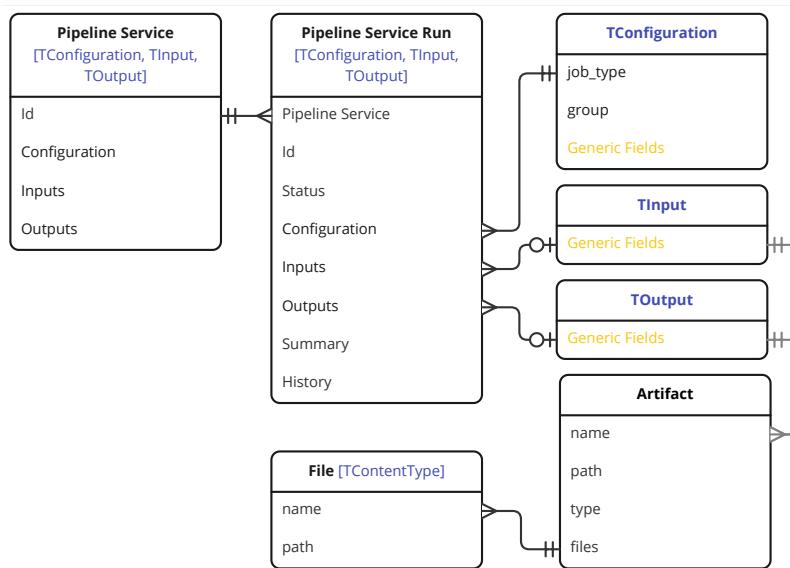
## 4.2 Modular Service-Based Framework

Certain tasks only consist of few processing steps, but others (e.g., sample generation) include more complex processing. For the latter, a minimal service framework has been developed. This framework provides generic building blocks that can be implemented and reused. The core components of the framework are outlined below.

- ▶ **Pipeline Service:** A service is a modular component responsible for performing a specific task. A service processes data, executes logic, and generates outputs or artifacts. The service lifecycle follows a pattern of initialization, processing, logging, and final output.

- ▶ **Pipeline Service Run:** When a service is executed, it creates a “run” that processes the input, applies the configuration, and generates the corresponding output. Runs provide insights into how a service is executed, and the results can be tracked and analyzed.
- ▶ **Configuration:** Each service requires specific configurations that define how it operates. This could include settings for compute resources, hyperparameters, or specific actions that need to be performed.
- ▶ **Inputs & Outputs:** Services take clearly defined inputs and generate specific outputs. Inputs and outputs represent the data processed by a service, usually as artifacts, though other data types can also be used.
- ▶ **Artifact:** An artifact refers to a versioned, trackable data object, such as datasets, models, or other files, used or produced during service execution.
- ▶ **File:** A file is a structured data object containing content in various formats (e.g., text or audio). Files are part of an artifact, making them accessible and trackable.

**Generic Data Types:** Figure 4.2 illustrates the framework’s components and the use of generic data types. When defining a new service using this framework, it inherits from `PipelineService`, requiring the implementation of abstract methods and the specification of generic data models (`TConfiguration`, `TInput`, and `TOutput`). Rather than using plain dictionaries, this employs clearly defined types for related configuration and processed data. Typed data models offer advantages over plain dictionaries as they provide structure, type safety, and validation. Explicitly typed models prevent errors by ensuring correct data formats and types at runtime. This reduces bugs arising from missing or misconfigured settings. Moreover, typed models offer enhanced coding assistance, such as autocompletion and type hints.



**Figure 4.2:** Pipeline Service and Execution ERD. The diagram shows the relations of services with generic configuration, inputs, and outputs (`TConfiguration`, `TInput`, `TOutput`). Each service run is tracked by a Pipeline Service Run, which records the service’s execution details. The configuration defines how the service operates, while the input and output types represent the data (e.g. artifacts) it processes.

2: Python library for data validation: <https://docs.pydantic.dev/latest/>

In this context, Pydantic<sup>2</sup> is used. Pydantic allows the creation of typed models with minimal boilerplate. It automatically parses and validates data models, ensuring type correctness and providing error details on validation failures. Pydantic supports complex data structures and nested models. Further, Pydantic simplifies JSON and YAML serialization and deserialization, allowing easy conversion between Python objects and different data formats.

**Abstract Processing Method:** The `PipelineService` class provides core functionalities, but leaves the actual processing logic abstract. The `process` method, which subclasses must implement, receives a `PipelineServiceRun` object containing execution information such as configuration and inputs. The method is responsible for performing the service's machine learning operations, and the results are written back to the `PipelineServiceRun` outputs. This structure allows the script executing the service to access all necessary information for a complete and traceable service run.

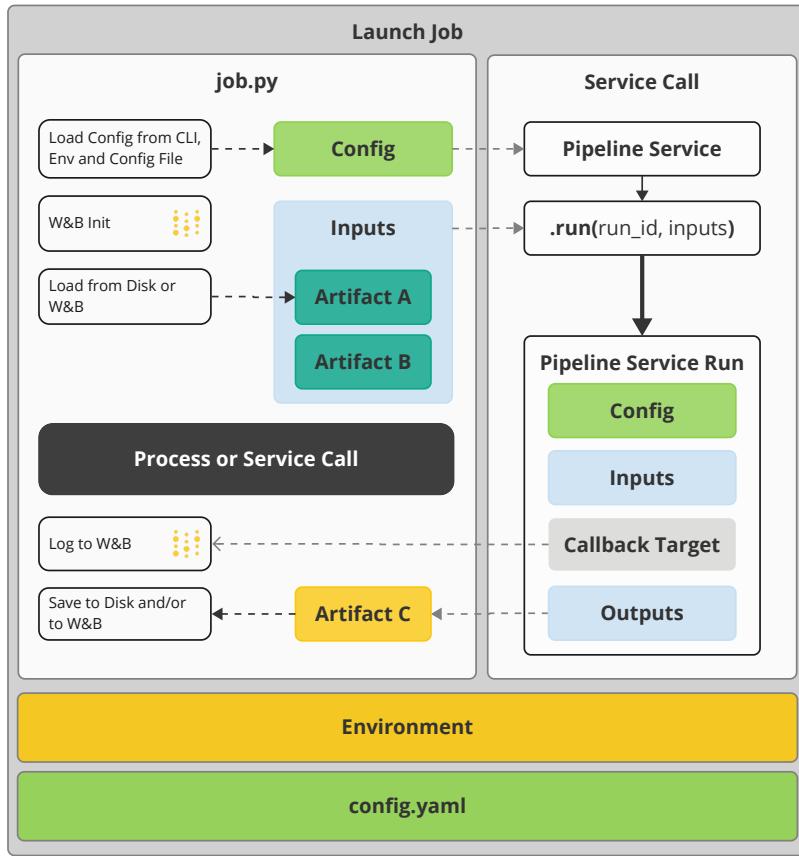
**Local Logging:** The framework's local logging tracks execution by storing incremental updates in a structured `summary.json` file. Each `PipelineServiceRun` can resume from previous states, update configurations dynamically, and maintain a step-by-step history for traceability. Artifacts are stored in a standardized structure for easy storage and access.

### 4.3 Experiment Execution & Tracking

In 4.2 services were introduced as code artifacts defining machine learning tasks. This section outlines the execution of these services, including tracking key metrics, configurations, and outputs for reproducibility and transparency. To execute a machine learning task, both the code artifact and an environment are required. The environment specifies the execution context, including the interpreter and installed packages. A **job** combines a code artifact with a specification of the environment, along with the necessary inputs and configurations to run an experiment. Depending on the task's complexity, a job may invoke a service or directly define the processing logic. Larger tasks, such as sample generation or audio dataset creation, involve service calls, whereas simpler tasks, like dataset preparation, handle the processing inline.

3: Weights and Biases (W&B), a platform for managing machine learning experiments: <https://docs.wandb.ai/>

For remote tracking and logging, Weights and Biases (W&B)<sup>3</sup>, is used. While the service framework already supports local logging, W&B enables remote result tracking and provides additional features. Services generate local outputs and maintain logging histories independently, while still leveraging remote tracking when needed. Consequently, a job defines not only how experiments are processed but also the remote logging setup, including configurations, key metrics, history, and artifact usage and production.



**Figure 4.3:** Functionality and Structure of a Launch Job. A Launch job integrates a code artifact with the environment specifications, inputs, and configurations needed to execute a machine learning experiment. The executed code typically involves service invocation and defines inputs loaded from, and outputs passed to W&B.

**Launch Job:** Figure 4.3 illustrates the concept of a launch job. A launch job integrates the code artifact with the environment required to execute an experiment. Experiments in this work were run either locally using a virtual environment or on a remote cluster using SLURM. For SLURM-based runs, the launch job also includes batch job definitions. The SLURM batch jobs clone the code from GitHub, check out the specified branch or commit, and install the required dependencies for each launch job.

A typical job definition starts by loading configuration through the `from_cli` method, which merges default values, YAML file contents, and command-line arguments, with command-line inputs taking priority. This allows configuration to be defined at three levels: environment variables (as default values), YAML files, and command-line arguments. After loading the configuration, W&B is initialized, and the configuration is passed to it. Jobs that require input artifacts load them either from the local disk or from W&B. Sensitive artifacts are stored locally, while a placeholder with metadata about their location is logged to W&B to ensure data lineage.

The job processes data based on the provided configuration and inputs. The processing logic is either defined directly within the job or as a service. In the latter case, the service is called with the configuration, and inputs are passed using the `run` method, which returns a service run. Upon completion, the service run

contains the final outputs and related information. The service logs locally by default and supports defining a callback target for remote logging with W&B. After processing, the resulting artifacts are logged locally and optionally on W&B to maintain artifact lineage.

**Data Lineage:** Data lineage in W&B enables clear tracking of the origins, transformations, and usage of datasets, models, and artifacts throughout an experiment’s lifecycle. By visualizing the artifact graph, dependencies can be traced, inputs and outputs identified, and reproducibility ensured. This transparency optimizes workflows and maintains consistency across experiments.

# Text Sample Generation

The first research question (RQ1) investigates how controlled text data can be generated to closely resemble children’s utterances and approximate the same distribution as their linguistic patterns. This chapter examines the use of LLM to generate realistic text data, including learner errors, which is then converted to audio (see Section 6) and used as additional training data (see Section 7). This approach aligns with the concept of data augmentation, where existing data is leveraged to generate additional training data, reducing the need for collecting new data (Feng et al., 2021). Although using LLMs for data augmentation can produce realistic training data, the effectiveness of data generation depends on factors such as prompt design, task complexity, and the quality, quantity, and diversity of the generated data.

5.1	Methods	37
5.2	Results and Deliverables	42
5.3	Conclusion And Future Work	50

## 5.1 Methods

The ChaLL corpus (Section 4.1.1) consists of transcripts from Swiss children’s spontaneous language-learning conversations, with a subset enhanced by detailed ERRANT error annotations (Section 4.1.2). This experiment aims to replicate the corpus data while carefully controlling the generation and annotation of errors based on the annotated subset. To assess errors in the generated text, the methodology generates parallel sentences with grammatically correct and incorrect versions (Section 5.1.1). The process is structured as a sequence of LLM calls, guided by prompts specifically designed to align with ChaLL data (Section 5.1.2). ERRANT error types are filtered and justified as inputs to the sequence (Section 5.2.2), and the framework is implemented to generate the final synthetic text dataset (Section 5.1.3).

The selection of GPT-4o as the LLM for this experiment was driven by its advanced capabilities in text generation, making it well-suited for the task. While no extensive comparison of alternatives was conducted, GPT-4o offers state-of-the-art performance in text generation, producing semantically coherent and fluent texts (Islam & Moushi, 2024), and effectively incorporates nuanced grammatical patterns necessary for replicating children-like grammatical errors. Its large context window and modular prompt capabilities enable precise control over outputs, enabling the creation of controlled text samples. Additionally, its efficiency and scalability, with improved tokenization and faster processing speeds, make GPT-4o suitable for large-scale synthetic data generation at reduced costs (Islam & Moushi, 2024).

### 5.1.1 Generating Text Pairs

A key goal of the ChaLL project is to provide feedback on learners' errors, which requires the ASR system to accurately preserve errors in transcripts. While general accuracy, measured by WER, is important (e.g., for response generation), error preservation is critical for delivering effective corrective feedback and supporting ChaLL's role in improving second-language proficiency. Consequently, controlling and preserving errors is equally essential for generating realistic and effective training data.

To enable the controlled generation of specific error types, a data augmentation approach inspired by Park et al. (2024) is adopted. This approach uses an LLM to generate pairs of grammatically correct and incorrect sentences, ensuring realistic and diverse error patterns. While their method primarily aims to train GEC models with augmented data, in this context, the incorrect sentences are used for synthesis and model training. The correct sentences, on the other hand, serve as potential corrections and as counterparts for applying ERRANT error detection and classification.

Controlling text generation is essential to include specific error types. Generating sample pairs over simple texts allows for automatic evaluation using the ERRANT framework (Section 2.4.1) to align and annotate text pairs. This approach verifies whether the specified errors are present in the generated text. As there is no strict one-to-one relationship between incorrect and correct texts, each pair reflects one possible correction.

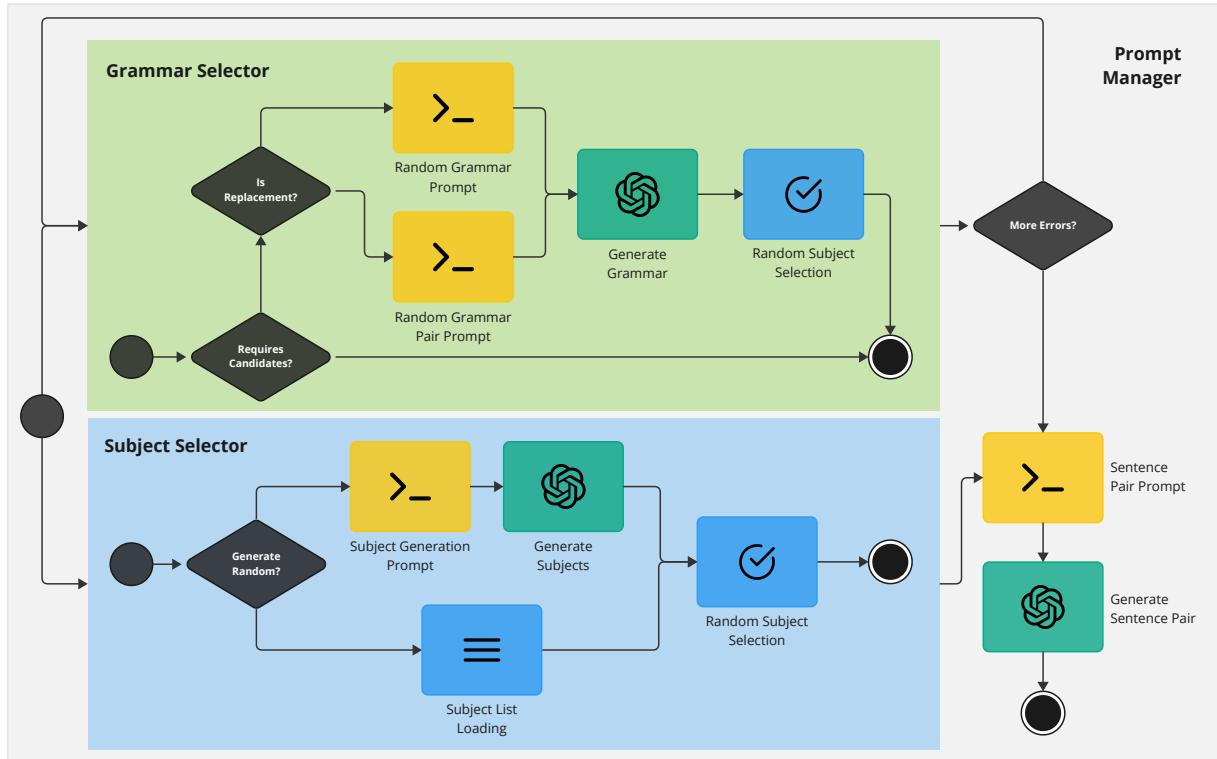
### 5.1.2 Sample Generation Chain

When generating language, an intuitive approach is to control all elements of speech. However, due to the infinite variability of language, it is impossible to fully control every element. Attempting to do so can even compromise the quality of the data (Park et al., 2024). Building on the approach proposed by Park et al. (2024), the hypothesis is that the primary factors determining the quality of generated pairs are the types of subjects and grammatical errors.

Following the approach of Park et al. (2024), the process is structured as a chain of prompts rather than relying on a single prompt. But it differs from the ChatLang framework in its structure, the number of errors generated per sample pair, and the specific prompts used to create the data. The chain includes the Subject Selector (Section 5.1.2.1), Grammar Selector (Section 5.1.2.2), and Prompt Manager (Section 5.1.2.3) as illustrated in Figure 5.1 and summarized in Algorithm A.2.1. The prompts and the chain are developed iteratively with incremental updates. LangChain<sup>1</sup> was used to create modular prompt chains with reusable components for targeted tasks, while LangSmith<sup>2</sup> enabled debugging, tracking, and iterative refinement.

1: <https://www.langchain.com/>

2: <https://www.langchain.com/langsmith>



**Figure 5.1:** Workflow of the sample generation process, illustrating how the Subject Selector and Grammar Selector contribute to generating subjects and grammar structures. These components feed into the Prompt Manger to produce the final response.

### 5.1.2.1 Subject Selector

A simple approach is to design prompts that generate pairs without considering the subject element. However, Park et al. (2024) has shown that this results in a limited variety of subjects. To address this, a **Subject Selector** is introduced to diversify subjects in parallel sentences. The idea of this component is to generate and select subject candidates for the final result. While initially using an LLM to generate subject candidates dynamically, the approach was later replaced with predefined subject lists as an alternative.

- ▶ **Generate Subjects Dynamically.** Based on a subject type, subject candidates equal to a specified window size (e.g., 30) are generated. From these, one candidate is randomly chosen as the sentence's subject. Details on Subject Generation Prompts can be found in Appendix A, Section A.4.1.
- ▶ **Predefined Subject List.** Here, the need to generate a subject for each sample is eliminated. Instead, a predefined list of subjects is used, with one randomly selected each time.

The predefined subject list is preferred for its lower computational cost and greater control, ensuring consistency and alignment with subjects related to CEFR A1-A2 levels. This approach avoids the overhead of dynamic generation while maintaining a diverse, controlled pool of subjects for sample creation.

3: <https://www.oxfordlearnersdictionaries.com/wordlists/oxford3000-5000>

**Subject List Generation.** The subject list was generated using the Oxford 3000 and 5000 word lists<sup>3</sup>, which provide essential vocabulary categorized by CEFR levels and parts of speech. The process focused on selecting nouns, particularly those at beginner levels (A1 and A2), to ensure relevance for subject creation. This base list was further expanded with additional categories generated using GPT-4o, including common names, cities, countries, landmarks, brands, sports teams, and fictional characters, resulting in a diverse pool of 1,497 subjects.

### 5.1.2.2 Grammar Selector

The issue of limited variety extends to grammar generation as well. A simple prompt designed to produce parallel sentences with a specific error type often results in a constrained set of errors. As highlighted by Park et al. (2024), focusing solely on a grammatical category (e.g., conjunctions) fails to address this limitation, frequently overusing common connectors such as “and” or “but”. To overcome this, the method employs a Grammar Selector, which generates a diverse set of candidates and then selects one randomly, ensuring greater variety in the grammatical structures used.

In contrast to Park et al. (2024), this work enables the generation of parallel sentences with multiple errors rather than limiting it to a single error. As illustrated in Figure 5.1, the Grammar Selector can be invoked multiple times to introduce multiple grammatical errors. It can also be configured to produce error-free sentences or variations with other differences, such as code-switching, depending on the settings. This approach offers greater control and flexibility over the generated grammar.

The Grammar Selector distinguishes between missing/unnecessary errors and replacement errors. For missing or unnecessary elements, a single grammar item (e.g., “but”) is generated, while for replacement errors (e.g., “but” → “and”) a pair of original and replaced elements is produced. In contrast, certain error types, such as Word Order (WO), are generated directly by the final Prompt Manager without prior grammar generation. These errors are effectively introduced by instructing the LLM to create them directly. Further details are provided in Section 5.2.2. Details on Grammar Generation Prompts can be found in Appendix A, Section A.4.2.

### 5.1.2.3 Prompt Manager

The Prompt Manager integrates subjects from the Subject Selector with grammatical errors from the Grammar Selector into a single prompt. The prompt is optimized to keep all elements identical except for the introduced grammatical differences. Additionally, the Prompt Manager controls other aspects of text generation:

- ▶ **Target Word Count:** Defines the length of the generated parallel texts, allowing control over text length.
- ▶ **Language Proficiency:** The prompt explicitly defines the CEFR A1–A2 level for the generated texts, specifying short, independent sentences with no complex clauses. This ensures linguistic simplicity and clarity, aligning with the patterns observed in children’s speech within the ChaLL corpus.
- ▶ **Code Switching:** If code-switching is enabled, the prompt instructs the LLM to replace English words with their German equivalents, suffixed by @g (e.g., “Haus@g”). This simulates real-world language mixing.

Details on the Prompt Manager Prompts can be found in Appendix A, Section A.4.2. The prompt is using the Jinja2<sup>4</sup> template. Jinja2 is used for dynamic replacing of placeholders, conditional logic (if statements) to handle cases like missing error rules or enabling code switching, and most importantly for loops to iterate through error lists and generate instructions for each. This approach allows the prompt to adapt to varying inputs and conditions, ensuring flexibility and precision. The result is a structured output formatted as a JSON instance.

4: Jinja2 is a templating engine for Python that allows dynamic content generation using placeholders, loops, and conditional statements, making templates more flexible and powerful: <https://jinja.palletsprojects.com/en/stable/>

### 5.1.3 Dataset Generation

Finally, the framework is used to generate synthetic text data that is synthesized and used as training data. The goal is to use 60 hours of the ChaLL data for training and the rest for evaluation and testing. With 60 hours of real training data available, the target is to generate 100 hours of synthetic audio. This allows for training and evaluating different proportions of real and synthetic data.

To determine the number of synthetic text samples required to generate 100 hours (360'000 seconds) of audio, the average audio durations and word counts from ChaLL samples are used. For samples with at least two words, the average audio length is 5.43 seconds, and the average word count is 9.92 with a high standard deviation of 6.23. Thus, a target range of 4 to 16 words is defined for the word count. This corresponds to a mean expected word count (MEW) of 10, as the word counts are randomly selected within this range. Using an estimated speaking rate of 120 words per minute (WPM), the Expected Audio Duration (EAD) is calculated as:

$$\text{EAD} = \frac{60}{\text{WPM}} \times \text{MEW} = \frac{60}{120} \times 10 = 5 \text{ seconds.}$$

Given the target duration of 360'000 seconds, a total of approximately 72'000 generated samples are required. This number, however, can be reduced by synthesizing the same text sample with multiple voices. To balance speaker diversity and cost considerations, a total of 34,000 samples are generated and synthesized with two different speakers voices.

To generate the required data, the process was divided into three chunks of 12,000 samples each. Input data was further processed in smaller batches to optimize efficiency and manage resources effectively. The batch size was dynamically determined based on the user’s API tier and total sample size. Rate limits are handled using a retry mechanism, introducing incremental delays for failed requests, up to a predefined maximum retry count.

## 5.2 Results and Deliverables

The objective of this chapter is to create a corpus of realistic speech that replicates the characteristics of the ChaLL data. Before presenting the final artifact in Section 5.2.4, this chapter first discusses the evaluation of the ERRANT-annotated ChaLL subset (Section 5.2.1) and the selection of ERRANT-based error types (Section 5.2.2).

### 5.2.1 ERRANT-annotated ChaLL Subset Evaluation

Figure A.1 in Appendix A presents the frequency of the top 30 error types. The top 30 includes only error codes that appear at least twice in the annotated subset. The top panel represents all human-annotated errors with valid ERRANT error codes. The bottom panel excludes errors arising from code-switching. Without code-switching errors, the most frequent category is replacement (284), followed by missing (67) and unnecessary (61) errors. Table 5.1 further breaks down these counts by category and tier for non-code-switching errors.

**Table 5.1:** Distribution of errors across three linguistic tiers: **Morph** (morphological errors), **Other** (uncategorized errors), and **POS** (part-of-speech errors). Errors are categorized as **M** (Missing), **R** (Replacement), or **U** (Unnecessary), with row and column totals.

Type	M	R	U	Total
Morph	2	53	2	57
Other	0	97	0	97
POS	65	134	59	258
Total	67	284	61	412

Most student errors are replacement errors. In the “Other” category, the most frequent errors are word order errors (“R:WO”, 64), which is the most common error code overall, followed by spelling errors (“R:SPELL”, 29). Morphological errors are dominated by noun number errors (“R:NOUN:NUM”, 21), verb agreement errors (“R:VERB:SVA”, 20), and verb form errors (“R:VERB:FORM”, 4).

The POS category is also primarily composed of replacement errors but includes some missing and unnecessary errors. Most POS replacement errors are accounted for by prepositions (“R:PREP”, 33), verbs (“R:VERB”, 24), and nouns (“R:NOUN”, 22). The most frequent unnecessary error code is determiner (“U:DET”, 21), while the most common missing errors are verb (“M:VERB”, 14), other (“M:OTHER”, 13), and pronoun (“M:PRON”, 11).

These results highlight that most errors in children’s spontaneous speech are replacement errors, with frequent issues in word order (“R:WO”) and prepositions (“R:PREP”) indicating challenges in structuring speech accurately in real-time. Morphological errors, such as noun number and verb agreement, reflect additional difficulties in spontaneous grammatical processing. The lower frequency of missing and unnecessary errors suggests that while speakers attempt to include all necessary elements, they often struggle with accurate word selection and structure.

Error Code	Text
R:WO	Okay. Uhm can have <b>you</b> ice cream ?
R:PREP	Yes, it's <b>of</b> this side.
R:SPELL	Two <b>sings</b> in her hand?
R:VERB	<b>I</b> s has the girl green hair ?
R:NOUN	Is the girl on the <b>round</b> ?
R:NOUN:NUM	It's uhm it is giftig. It is blue mit red <b>stripe</b> and eating .
U:DET	Excuse me. Do you have <b>a</b> popcorn?
R:VERB:SVA	Uhm. The girl in my picture <b>have</b> brun hair.
R:PRON	In the background I have a poster <b>who</b> is yellow with a girl on it .
M:VERB	Nice. Um @! you have a window?
M:PRON	Yes, it's big. @! is a castle.
M:DET	It is @! banana.
U:PREP	I can see uh <b>at</b> the couch.
R:ADJ	Cool. I likes <b>changy</b> colours. And I hearts swimming.

**Table 5.2:** Selected samples for the 14 most frequent error codes. Each row highlights the respective error type in bold, while other errors in the text remain unmarked. Missing words are marked with “@!”.

Table 5.2 provides examples of the 14 most frequent error codes. Each row highlights the respective error type in bold within a sample sentence, while other errors in the text are not marked. These examples illustrate typical patterns and contexts in which these errors occur.

## 5.2.2 Error Type Selection and Definition

This work leverages the ERRANT framework to define error types for sample generation and evaluate error preservation. ERRANT’s standardized error definitions are used to generate erroneous text samples. Since not all error types are feasible in this context, the selection is guided by two factors: applicability to this work and prevalence in the human-annotated subset.

### 5.2.2.1 Impact of ERRANT Error Categories

To determine the applicability of error categories, it is crucial to understand their impact on the synthesized text. **Replacements (R)**

involve substituting a word in the correct sentence (e.g., replacing “a” with “the”), resulting in audio and transcription reflecting the altered word, which grammar correction should identify and fix. **Missing (M)** errors omit a word or phrase, leading to incomplete sentences in the audio, requiring the missing element to be inserted during transcript correction. **Unnecessary (U)** errors add extraneous words that disrupt grammar or context, requiring their detection and removal to restore correctness.

The correct text represents one possible correction of the corresponding incorrect text. By applying ERRANT error annotation between these versions, the differences are identified and categorized. In this context, the focus is not on the specific differences themselves (e.g., “a” -> “the”) but on the type of change they represent (such as R:DET).

For these three error categories (R, M, U), specific conditions must render the text ungrammatical. In the case of **replacements**, the sentence is only incorrect if the substituted word is not an appropriate alternative. Some word types, such as determiners, conjunctions, and particles, are more likely to cause grammatical errors when replaced. In contrast, other word types, like adjectives, adverbs, nouns, and verbs, are often more flexible and can be substituted without making the sentence ungrammatical.

For **missing** errors, the text becomes ungrammatical when the omitted word or phrase is critical to the structure or meaning of the sentence. Functional words, such as determiners, auxiliary verbs, and prepositions, tend to cause significant grammatical issues when omitted. However, omitting less critical elements, such as modifiers, may not always lead to ungrammaticality.

For **unnecessary** errors, the text becomes ungrammatical when the introduced word or phrase disrupts the sentence’s grammar or logical coherence. This is particularly common with extra determiners, prepositions, or conjunctions, which can conflict with the sentence’s existing structure. In contrast, adding redundant adjectives or adverbs may not always result in ungrammatical text but can still make the sentence awkward or contextually inappropriate.

### 5.2.2.2 Excluded ERRANT Error Types

The selection of error types from the 55 ERRANT categories is summarized in Table A.1 in Appendix A. Error codes are color-coded for clarity: green indicates errors that are used, red indicates those that are not used, and orange marks errors that are used with caution (e.g., because they represent semantic errors). This categorization is based on the most common errors observed in the UZH dataset.

Certain ERRANT error types are excluded from sample generation due to their reliance on the TTS system’s ability to handle specific

elements like apostrophes, capitalization, whitespace, and punctuation marks. The **Other** category is omitted for its lack of clear definition. **Contractions** (e.g., “it’s” vs. “it is”), **Orthography** (e.g., case or whitespace errors), **Noun Possessive** (e.g., “John’s book”), and **Punctuation** (e.g., misplaced commas or periods) errors are excluded as their correctness depends heavily on the TTS system’s capabilities, which is beyond the scope of this study.

Further, the error types **M:ADJ**, **U:ADJ**, **M:ADV**, and **U:ADV** are excluded. The omission of adjectives or adverbs typically does not render a sentence ungrammatical but rather affects its completeness or descriptiveness. For example, a sentence like “The house is.” may lack clarity without an adjective but remains grammatically correct. Similarly, unnecessary adverbs or adverbs add redundancy without breaking grammatical rules (e.g., “She ran quickly fast”).

Additionally, **M:CONJ** is excluded because its omission often results in the STT system treating the omission as a sentence boundary, inserting a period instead. This can produce sentences that are either perfectly grammatical or introduce a missing word error. For example, “He reads a book and eats spaghetti” might become “He reads a book. @! eats spaghetti,” where the missing conjunction is treated as a structural issue rather than a purely grammatical one. While these elements contribute to sentence completeness or nuance, they are not considered in this study’s scope.

Orange-coded error types primarily represent **semantic errors**. For semantic errors, such as “my loved item” instead of “my favourite item,” these are annotated with an “R” tag and the corresponding word category (in this case, ADJ). This approach treats semantic errors in the same manner as lexical errors.

### 5.2.2.3 Selected ERRANT Error Code Inputs

The ERRANT error codes used as inputs for sample generation must be specified by description and example/s, which guide the Grammar Selector in generating candidates. Table A.2, located in Appendix A, provides a detailed overview of these error codes with their corresponding descriptions and examples. The error codes without descriptions are used to generate simple grammar rather than pairs, following the prompt detailed in Section A.4.2.1. The descriptions and examples are derived from ERRANT (Bryant et al., 2017) and are validated to ensure appropriate grammar generation.

## 5.2.3 Sample Generation App

The Sample Generation App (see screenshot in Appendix A.2), built with Gradio<sup>5</sup>, offers an interactive platform for exploring the

5: <https://www.gradio.app/>

sample generation. Users can customize key parameters such as subject type, number of errors, target word count, and whether to include code-switching. The app dynamically generates text pairs based on these inputs, applying specified grammatical error rules while maintaining parallel structure. Outputs include the generated texts, JSON-formatted results, and visual error annotations using ERRANT. This interface provides an accessible way to experiment with the framework. During the development of the framework, this tool was instrumental in refining prompts and logic, allowing for iterative improvements and better alignment with the desired outputs.

### 5.2.4 Synthetic Text Dataset

The data generation process successfully processed **34,000** samples through 90,895 successful requests, using a total of 18,191,723 tokens. This included 11,636,675 prompt tokens and 6,555,048 completion tokens, with a total cost of \$156.51. The entire generation was completed in 10 hours, and 45 minutes.

Key features include `text` (the grammatically correct version) and `corrected_text` (the grammatically incorrect version). Additional metadata includes `spans` (specific spans related to errors), `target_word_count` (desired word count), `use_code_switching` (whether code-switching is applied), `subject` and `subject_type` (providing context for the text), and `errors` (details about the grammatical issues introduced).

#### 5.2.4.1 Qualitative Text Dataset Evaluation

Table 5.3 provides an overview of selected samples from the generated dataset, illustrating the relationship between configured errors (Error Config) and those detected by ERRANT (Error Spans). It highlights original and corrected texts, emphasizing differences in error types and spans to demonstrate the effectiveness of the error generation and detection process. For example, in the sample “He careful chose the route, but he refused participate” the configured and detected errors (R:MORPH, M:VERB:FORM) align perfectly, reflecting accurate error generation and detection.

However, discrepancies are observed in different cases. In the sample “Hello Kitty is a cute character. Her run a school to to learn English,” the configured errors (R:VERB:TENSE, U:VERB:FORM, R:PRON) partially match the detected errors, where ERRANT misses R:PRON and detects an additional error labeled as R:OTHER. For samples with code-switching as in “Mittwoch@g is the middle of the week”, ERRANT detects R:NOUN, highlighting the code-switched German word.

**Table 5.3:** Overview of selected samples from the generated dataset. Error Config refers to the configured errors used as input during generation, while Error Spans indicate errors detected by ERRANT. The table highlights original and corrected texts, showcasing differences in error types and spans.

Text (Original & Corrected)	Error Config	Error Spans
<p><b>O:</b> The these water level in the pool is swim high. We can swim today.</p> <p><b>C:</b> The water level in the pool is high. We can swim today.</p>	U:DET, U:VERB	U:DET, U:ADJ
<p><b>O:</b> Nyon is a small city in Switzerland. It is near Lake Geneva.</p> <p><b>C:</b> Nyon is a small city in Switzerland. It is near Lake Geneva.</p>	None	None
<p><b>O:</b> He careful chose the route, but he refused participate.</p> <p><b>C:</b> He carefully chose the route, but he refused to participate.</p>	R:MORPH, M:VERB:FORM	R:MORPH, M:VERB:FORM
<p><b>O:</b> Hello Kitty is a cute character. Her run a school to to learn English.</p> <p><b>C:</b> Hello Kitty is a cute character. She has run a school to learn English.</p>	R:VERB:TENSE, U:VERB:FORM, R:PRON	R:OTHER, U:VERB:FORM
<p><b>O:</b> Winner ran the finish line.</p> <p><b>C:</b> Your winner ran across the finish line.</p>	M:DET, M:PART	M:DET, M:PREP
<p><b>O:</b> Mittwoch@g is the middle of the week. I like Wednesdays.</p> <p><b>C:</b> Wednesday is the middle of the week. I like Wednesdays.</p>	None	R:NOUN
<p><b>O:</b> Finland compose many beautiful lakes stunning natural scenery.</p> <p><b>C:</b> Finland comprises many beautiful lakes with stunning natural scenery.</p>	M:PART, R:VERB	R:VERB, M:PREP

#### 5.2.4.2 Quantitative Text Dataset Evaluation

**Code-Switching, Subjects, and Word Counts Distributions:** Figure A.3 in Appendix A.2 includes four plots showing distributions of code-switching, subjects, and target word counts. Most cases show code-switching annotations (@g) present, though this only confirms the annotation’s inclusion, not the accuracy of code-switching. Subject presence is determined by matching subject words in both correct and incorrect texts, with subjects typically included in generated samples. Code-switching distribution aligns with the configured 20% usage rate, while target word counts show a peak at 16, reflecting the first batch’s fixed target of 16 words. Subsequent samples are generated with random word counts (4–16) based on error count. Figure A.4 depicts the relationship between configured number of errors and number of errors detected using ERRANT. Generally, detected errors align with configured values, but discrepancies exist. For lower configured errors, more errors tend to be detected, whereas higher configured number of errors often result in fewer detected errors.

**Discrepancies in Generation and Annotation:** Table A.3 summarizes the total counts of errors defined as inputs during the generation process and those detected by the ERRANT annotation tool. The “Error Count” column indicates the number of times an error was configured for generation, with a value of 0 for excluded error codes and evenly distributed counts for others, as error types are randomly selected. The “Span Count” column shows how often ERRANT detected errors by comparing the incorrect and correct text versions, extracting edits, and classifying them into the error codes. The last column highlights discrepancies between configured and detected errors, revealing significant differences for certain error codes.

The results highlight substantial discrepancies, particularly for error codes that are not explicitly configured but still detected in significant numbers, such as U:ADV (2033 spans) and R:OTHER (2647 spans), the latter being a catch-all category for errors that did not fit any specific classification. In contrast, for some error types that are used, the detected spans were either lower (e.g., R:PREP, -1456 difference) or higher (e.g., R:PART, +1192 difference) than expected. This suggests inconsistencies in error generation, detection and classification, potentially due to misalignments in annotation or overlapping error categories. Additionally, notable positive differences, such as R:ADJ:FORM (+1161 spans) and U:CONJ (+935 spans), indicate cases where the annotation tool detected more instances than configured, possibly reflecting systematic overcorrection.

**Analysis of Error Alignment:** To better understand those differences, Figure A.5 analyzes the relationship between configured and ERRANT-detected errors through sample-wise list comparisons of input error codes and detected error codes. Three outcomes are possible: A defined input error is correctly annotated by ERRANT (**Hits**). A defined input error is not annotated by ERRANT (**Misses**). An error not defined as an input is annotated by ERRANT (**Extraneous**). The figure highlights the most common error codes for each of these outcomes.

The **Hits** distribution shows that commonly introduced errors, such as M:VERB:FORM, R:VERB:FORM, and M:DET, are frequently detected correctly, indicating reliable generation and annotation for these categories. In contrast, the **Misses** category highlights errors that were defined but either not generated as expected or not recognized, with M:PART, U:PART, and R:PART among the most frequently overlooked, suggesting challenges in either generating these errors or detecting them correctly. The **Extraneous** errors category is dominated by R:OTHER, which serves as a fallback classification when no specific category applies, along with U:ADV and R:PREP, indicating a tendency for ERRANT to over-annotate certain structures or for the model to generate unintended errors. Notably, categories such as R:SPELL and U:PREP appear in both Misses and Extraneous groups, hinting at inconsistencies that

may stem from either error generation or classification. These findings suggest that while ERRANT performs well for certain error types, others suffer from misalignment due to misgeneration, misclassification, omissions, or over-detection, underscoring the need for further refinements in both the error generation process and automated error annotation.

**Error-Type Confusion:** Finally, Figure A.6 shows a confusion matrix for configured and detected errors, highlighting common error-type confusions during generation and annotation. Since multiple errors can be configured for a single sample, configured and detected errors cannot be directly aligned. Instead, all possible combinations of input and detected error codes are analyzed. To focus on the most significant relationships, only error pairs with at least 200 occurrences are included in the matrix. The matrix reveals notable patterns. For example, “M:DET” and “R:MORPH” frequently align, suggesting a strong correlation between determiner errors and morphological adjustments in annotation. Similarly, “R:NOUN:INFL” and “R:NOUN:NUM” show significant overlaps, reflecting challenges in distinguishing noun inflection and pluralization errors. Additionally, “U:PREP” and “R:PREP” often coincide, indicating the annotation process commonly detects errors related to prepositions, irrespective of whether they are explicitly configured.

#### 5.2.4.3 Performance Metrics for Error Detection and Annotation

To conclude the evaluation of the text dataset, its overall quality is assessed using three key performance metrics: **Precision**, **Recall**, and **F1-Score**. These metrics quantify the degree of alignment between the configured errors and those detected by ERRANT, providing an objective measure of generation and annotation accuracy.

Precision quantifies the accuracy of detected spans by measuring the proportion of correctly detected spans (Hits) among all generated spans, including extraneous ones. A high precision indicates that most detected errors align with the intended configurations:

$$\text{Precision} = \frac{\text{Hits}}{\text{Hits} + \text{Extraneous}} \quad (5.1)$$

Recall evaluates the coverage of the annotation process by measuring the proportion of correctly detected spans (Hits) out of all spans configured for generation. A high recall indicates that most configured errors are successfully detected:

$$\text{Recall} = \frac{\text{Hits}}{\text{Hits} + \text{Misses}} \quad (5.2)$$

F1-Score provides a balanced measure that combines precision and recall as their harmonic mean. It reflects the overall performance of the error generation and detection process, balancing accuracy and coverage:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5.3)$$

The performance metrics in Table 5.4 reflect the combined effects of both error generation and detection. A Precision of 0.608 indicates that most detected errors align with the intended configurations, though some extraneous errors suggest inconsistencies in either the generation process or over-detection by ERRANT. The Recall of 0.588 shows that a substantial portion of configured errors were successfully detected, but some were either not generated as expected or missed during annotation. The F1-Score of 0.594, balancing precision and recall, underscores the interplay between generation quality and annotation accuracy. These results highlight the need for refinements in both generating controlled errors and accurately detecting them.

**Table 5.4:** Performance Metrics for Text Dataset Evaluation

Metric	Value
Precision	0.608
Recall	0.588
F1-Score	0.594

To put these scores into perspective, Table C.1 in Appendix C presents the performance of UZH annotations compared to automatically detected errors using the approach from Section 7.1.3. Generating a corrected version with GEC and annotating both using ERRANT results in significantly lower scores. While these performances are not directly comparable, they highlight the challenge of accurately generating and detecting errors.

### 5.3 Conclusion And Future Work

This chapter demonstrated how LLMs can be used to generate synthetic text data replicating real learner's speech including errors, contributing to data augmentation efforts. The findings highlight the importance of prompt design in producing effective training data. Given the first objective the aim was to assess and validate the ability of LLMs to generate datasets. While this requires ongoing research, the proposed approach has shown how controlled samples including learners' errors can be generated. The sample generation process is designed as a sequence of operations, enabling greater flexibility and variability in generating subjects and errors. By integrating these features into the prompt manager alongside overall sample characteristics, child-like speech samples with specific error types can be generated.

The proposed approach highlights the challenges and limitations of generating accurate language. While GPT-4o is effective at following instructions, including too many details or attempting to control all aspects of the output often results in inaccuracies. Overloading a single prompt with multiple errors or excessive characteristic details can hinder the quality of the generation. But generating spontaneous, low-proficiency child-like language including hesitations, repetitions, reformulations, sentence breaks, code-switching, and mistakes requires a lot of instructions. Thus, for simplicity and to not overload the LLM with instructions, the focus was primary on learners' mistake. Additionally, code-switching has been incorporated as an optional feature during generation. However, the prompts or chain could be adapted to introduce disfluencies during generation. The language proficiency (A1-A2) and simplicity of the texts (short, independent sentences; no complex clauses), on the other hand, have been considered in the prompt.

Text pairs were preferred over simple texts, as parallel sentences can be annotated using tools like ERRANT. Comparing input error types with ERRANT-annotated errors offers an evaluation criterion, but this approach has limitations. The evaluation revealed low scores in precision and recall. Whether these inaccuracies are due to the generation or classification cannot be fully determined. But ERRANT tends to mix up certain types or bundles multiple errors. The rule-based ERRANT tool could potentially be replaced with a neural-based alternative or even a LLMs for improved performance in future work. While the ERRANT approach is proposed as an evaluation criterion, the generated text sample dataset has not been filtered based on it.

In addition to improving prompts and the chain, the inputs also need to be reassessed. This includes the choice of error types and the associated descriptions and examples. Error types with low evaluation scores, such as U:ADV and R:OTHER, should be updated or removed to address inconsistencies.

Furthermore, this work does not include an extensive comparison of different LLMs. While the generation and instruction capabilities are proven to be precise, other systems may perform equally well. Using an open-source LLM could significantly reduce sample generation costs. Alternatively, switching to GPT's batch API could cut costs by 50% and increase rate limits. However, LangChain does not natively support batch processing, which is why it was not implemented in this work.

Human evaluation was not conducted for this objective. However, to fully assess the capabilities of the sample generation framework (and error classification), human evaluation must be performed on the generated results. But overall, the proposed framework has proven how to control text samples that replicate the spontaneous second-language speech patterns observed in the ChaLL corpus, including learners' errors.

# 6

# Audio Generation

6.1	Methods	52
6.2	Results and Deliverables	57
6.3	Conclusion and Future Work	60

The second (SRQ2) and third (SRQ3) research questions address the selection of an optimal TTS system and the generation of an augmented audio dataset. The first objective is to evaluate various TTS systems in synthesizing text data from the ChaLL corpus. This involves comparing real and synthetic audio of the same transcripts to assess speaker similarity and naturalness through subjective evaluations, complemented by DataSpeech for automated analysis. The goal is to identify the TTS system that delivers the highest levels of similarity and naturalness.

Using the selected TTS system with voice cloning capabilities, the second objective focuses on generating a synthetic speech dataset. Erroneous samples from Chapter 5 serve as text prompts and are synthesized to replicate authentic speech patterns using real children’s voice data. This augmented dataset is subsequently utilized as additional training data in Chapter 7.

## 6.1 Methods

To generate the synthetic speech dataset (Section 6.1.4), potential speaker audio samples are first filtered and selected as voice references for voice cloning (Section 6.1.1). These references are used for both dataset generation and system evaluation. The TTS systems are compared using subjective metrics for speaker similarity and naturalness, supplemented by an automated evaluation through DataSpeech (Section 6.1.2).

Code-switching (with German or Swiss-German) is not inherently supported by the evaluated TTS systems. Tests were conducted to assess the systems’ potential for code-switching, and while the results outlined in Section 6.2.1 are promising, for the sake of simplicity, this feature was not prioritized in the final evaluation and dataset generation.

### 6.1.1 Definition of Voice References

The ChaLL dataset contains audio recordings of spontaneous English speech from Swiss children (grades 4–6), collected during language learning tasks to elicit authentic spontaneous speech. It includes transcriptions with annotations of learners’ errors. The same data is now being repurposed for voice conversion. However, not all samples and speakers are suitable for this purpose, so a subset of voice reference samples is first defined.

**Load ChaLL Dataset.** The ChaLL dataset is accessed via Huggingface<sup>1</sup>. As a Huggingface dataset, it supports various experimental setups through distinct builder configurations. The `asr_acl` configuration, defined in the prior study (Michot et al., 2024), preprocesses data by segmenting audio, limiting chunk length to 12 seconds, and removing trailing pauses to optimize it for ASR tasks. Each chunk contains speech from a single speaker, ensuring clear separation and manageable lengths. While speakers may appear in multiple original audio files, they cannot be linked across occurrences. The same configuration is used here, but slightly adapted: All data is loaded into a single split, with texts retaining its original casing and punctuation to preserve prosodic and structural features essential for the TTS system. After loading, the dataset is randomly shuffled to ensure unbiased selection.

**Filter Samples.** Not all samples and speakers from the original dataset are suitable for voice conversion. To ensure relevance, samples are filtered based on audio duration, word count (excluding fillers like “uh” and “um”), and speaker sample sufficiency. The process is managed through configurable settings and includes:

1. Filtering samples by duration (between 4 and 12 seconds) and word count (4).
2. Retaining samples of speakers with a minimum number (20) of valid samples.
3. Limiting the total number of samples per speaker (20).

The configuration used is shown above in brackets. The resulting voice reference dataset retains the same fields as the original ChaLL data and consists of **7780 samples** from **389 speakers**, based on the specified criteria. Note that the number of speakers does not directly match the participants in the ChaLL corpora, as some participants are excluded during filtering, and others may appear as distinct speakers since cross-file speaker linking is not possible.

### 6.1.2 TTS System Comparison

After selecting suitable voice references, the next step is evaluating and selecting an appropriate TTS system based on their ability to replicate natural and speaker-similar synthetic speech. Three systems are selected for detailed evaluation: Coqui TTS (Section 2.2.5), F5 TTS (Section 2.2.4), and E2 TTS (Section 2.2.3).

Other systems, such as MARS5<sup>2</sup> and GPT-SoVITS<sup>3</sup>, were also considered but not included in the evaluation. MARS5-TTS is excluded due to its very slow inference time, making it impractical for generating the 100 hours of data required. GPT-SoVITS was excluded because its voice cloning performance relies on finetuning. While finetuning with only 5 minutes of audio yields impressive results, it does not align with the study’s requirement for broader generalization.

1: <https://huggingface.co/datasets/mict-zhaw/chall>. Due to the sensitive nature of data involving minors, access is restricted and requires a collaboration agreement with the original project partners

2: Transformer-based model designed for high-quality speech synthesis: <https://huggingface.co/CAMB-AI/MARS5-TTS>

3: Zero-shot capable system leveraging GPT and voice conversion techniques: <https://github.com/RVC-Boss/GPT-SoVITS>

### 6.1.2.1 Subjective Evaluation

4: SMOS Scale:

- ▶ **1:** Completely Dissimilar.
- ▶ **2:** Mostly Dissimilar, with minor similarities.
- ▶ **3:** Somewhat Similar, noticeable differences in tone or style.
- ▶ **4:** Mostly Similar, minor differences only.
- ▶ **5:** Same Voice.

5: CMOS Scale:

- ▶ **-3:** Much worse than the reference.
- ▶ **-2:** Worse than the reference.
- ▶ **-1:** Slightly worse than the reference.
- ▶ **0:** Same as the reference.
- ▶ **+1:** Slightly better than the reference.
- ▶ **+2:** Better than the reference.
- ▶ **+3:** Much better than the reference.

The subjective evaluation aims to identify the most effective TTS system for generating synthetic speech samples with voice cloning capabilities. The selected system must accurately replicate the unique characteristics of young Swiss students speaking English as a second language. Human evaluation is essential to capture perceptual nuances, which automated metrics cannot fully assess, particularly for young learners with developing pronunciation and fluency. The evaluation process involves sampling speakers, generating audio with each system, and assessing the synthetic samples based on speaker similarity (SMOS<sup>4</sup>) and naturalness (CMOS<sup>5</sup>) as outlined in Algorithm 6.1.1.

9 specialists in linguistics or related fields rated each speech sample for SMOS and CMOS using a Gradio app (see Section 6.2.2 for details). The app is designed to be self-explanatory. Evaluators first receive instructions and are then guided step-by-step through the evaluation process. At each step, they are presented with a real audio sample from the ChaLL dataset and a synthetic audio generated using another sample from the same speaker to synthesise the transcript of the real audio. After listening to both, evaluators assign scores for both SMOS and CMOS.

#### Algorithm 6.1.1: Speaker Sampling and Evaluation

1. **Speaker Sampling:** Randomly sample speakers from dataset.
2. **Reference and Generation Sampling:** For each sampled speaker  $X_j$ :
  - ▶ Randomly sample a reference audio  $A_{\text{ref}}$  and corresponding reference text  $T_{\text{ref}}$ .
  - ▶ Randomly sample a new text  $T_{\text{gen}}$  for generation.
3. **TTS Generation:** For each TTS system  $i$  (where  $i = 1, 2, 3$ ), generate the audio  $A_{ij}^{\text{gen}}$  for speaker  $j$  using:

$$A_{ij}^{\text{gen}} = \text{TTS}_i(\text{ref\_audio} = A_{\text{ref}}, \text{ref\_text} = T_{\text{ref}}, \text{text} = T_{\text{gen}})$$

Here,  $A_{ij}^{\text{gen}}$  is the audio generated by TTS system  $i$  for speaker  $j$  based on the given references and generation text.

4. **Annotation:** In the annotation tool, the following items are presented for evaluation:
  - a) The generated audio  $A_{ij}^{\text{gen}}$ ,
  - b) The corresponding generation text  $T_{\text{gen}}$ ,
  - c) The reference audio  $A_{\text{ref}}$ .
5. **Rating:** Annotators rate the quality of the generated audio  $A_{ij}^{\text{gen}}$  based on...
  - a) SMOS (Similarity Mean Opinion Score)
  - b) CMOS (Comparative Mean Opinion Score)
6. **Final Evaluation:** Compute the mean ( $\mu$ ) and standard deviation ( $\sigma$ ) of the ratings for each TTS system. The system with the highest mean rating and lowest standard deviation is considered the most consistent and effective for voice cloning.

The evaluation scores from all evaluators were analyzed using statistical metrics to determine the overall performance of each TTS system. The mean and standard deviation of the SMOS and CMOS scores were calculated for each system, providing insights into both the average performance and variability in ratings<sup>6</sup>.

### 6.1.3 Automated Similarity and Naturalness Evaluation

The subjective TTS evaluation relies on human assessors who compare synthetic audio to ground truth speech. While human judgment remains the primary criterion for TTS system selection, this chapter introduces DataSpeech (Section 2.2.9) annotation as an automated complementary approach to measure similarity between reference speech and generated samples. DataSpeech systematically processes and analyzes audio datasets, extracting continuous features and keyword-based mappings to quantify phonetic similarity, prosodic patterns, and audio quality. By providing a reproducible and automated framework, it reduces subjectivity in human evaluation and enables scalable benchmarking.

For this evaluation, speech data generated from synthetic text prompts (like to one in the final in the final dataset) is used, unlike in subjective assessments where real texts are synthesized and compared. The approach compares the reference audio used during synthesis with the corresponding generated audio. A total of 400 samples from three TTS systems and the reference voice dataset are annotated using DataSpeech<sup>7</sup>.

A **simple linear regression** was then performed for each feature, using the reference audio’s annotations as the independent variable ( $X$ ) and the corresponding synthetic audio’s annotations as the dependent variable ( $y$ ). The regression is used to assess the linear relationship between the two sets of annotations. The coefficient of determination ( $R^2$ ), slope, and intercept are computed for each feature, quantifying the degree of alignment and systematic deviations between the synthetic and reference annotations. These results provide a statistical measure of how closely the synthetic annotations match the reference audio’s annotations.

### 6.1.4 Audio Dataset Generation

To simplify the generation of synthetic TTS audio, the Audio Generation service provides a modular end-to-end framework that dynamically selects a TTS model to synthesize input text using reference audio data (see Section 6.1.1). The process begins with dataset preparation, including shuffling, and pairing input text with reference audio to ensure structured and reproducible synthesis. The service then pre-processes (@-annotations removed, whitespace normalized, and proper punctuation) the text, retrieves the corresponding reference audio, and generates speech using

6: There is a minor design flaw here. Instead of directly creating a voice reference dataset with 20 speakers, a dataset with 60 speakers was generated to allow reuse in other experiments. The intention was for all testers to use the same samples by applying a static seed and using the same first 20 speakers. However, an implementation error occurred, and it was not considered that ‘set’ does not preserve the order of elements. Originally, the plan was to use MACE (Multi-Annotator Competence Estimation) (Hovy et al., 2013) to account for variations in annotator reliability and improve the robustness of evaluation results. However, due to the design flaw, this approach proved to have limited utility. While most samples were evaluated multiple times, they rarely received more than two scores, which is insufficient for MACE to effectively estimate annotator competence. Despite this limitation, the evaluation with MACE was conducted, and the best-performing system identified by MACE aligned with the top system based on the mean average score.

7: <https://github.com/huggingface/dataspeech>

**Algorithm 6.1.2**

1. **Initialize TTS Model:** Dynamically select and initialize the TTS model based on configuration.
2. **Prepare Datasets:**
  - ▶ Load the input and reference datasets and verify successful loading.
  - ▶ Retain only relevant columns (e.g., ID, text, speaker) in the input dataset.
  - ▶ Shuffle the input dataset by configured seed for diversity and reproducibility.
  - ▶ Build a lookup table **mapping reference audio IDs to dataset indices**.
3. **Iterate Through Input Samples:** For each input sample:
  - a) Retrieve the corresponding reference audio and text using the reference audio ID.
  - b) Preprocess the input text by cleaning annotations and normalizing whitespaces.
  - c) **Synthesize Audio:**
    - ▶ Generate audio using the TTS model with the input text and reference audio.
    - ▶ Save the synthesized audio as a '.wav' file with a unique identifier.
4. **Monitor Progress:** Log the number of processed and failed samples at regular intervals and log errors for failed samples.
5. **Finalize and Save Outputs:**
  - ▶ Cast audio in the dataset to the appropriate format for compatibility (by configured sampling\_rate).
  - ▶ Save the final dataset, including audio and metadata, to disk.

the selected TTS model. Synthesized audio is stored in the final dataset, including metadata. The process of generating synthetic audio is summarized in Algorithm 6.1.1.

The same process is used to generate audio for the subjective evaluation, DataSpeech analysis, and the final dataset. For subjective evaluation, real data from the TTS corpus is used as both text inputs and voice references. In other cases, synthetic text samples serve as inputs, while TTS data remains the voice references.

For the final audio dataset generation, the synthetic texts from Chapter 5 are synthesized using the evaluated TTS system and the voice references defined in Section 6.1.1. Audio is generated in four batches, covering two sample ranges (0–17,000 and 17,000–34,000) with two voices, ensuring each sample was synthesized twice for sufficient data. The process is run on a V100 GPU (32GB memory) per batch. Finally, the batches are consolidated into a structured, standardized dataset, ensuring compatibility with ChaLL data standards for training and further applications. This involves resampling audio, cleaning texts (e.g., removing annotations, converting numbers to words, normalizing case), and filtering samples to a maximum duration of 12 seconds. The audio durations were saved as metadata to facilitate training with different portions of real and synthetic data.

## 6.2 Results and Deliverables

Beyond the synthetic audio dataset (Section 6.2.5), this section presents additional results related to TTS selection and dataset generation. Code-switching capabilities were analyzed as a by-product (Section 6.2.1), and an evaluation app (Section 6.2.2) was developed to assess the generated audio. The TTS system comparison yielded subjective results using SMOS and CMOS (Section 6.2.3) and quantitative results through DataSpeech (Section 6.2.4) analysis.

### 6.2.1 Code-Switching Findings

This section presents additional findings that, while not central to this study, may be relevant for future research. A key challenge for language learning apps is handling code-switching, such as when children mix their mother tongue (Swiss-German) with English. This chapter summarizes two approaches explored to enable code-switching with F5 and Coqui TTS.

To enable code-switching with **F5 TTS**, a fine-tuned version of the F5 model trained on German data was used<sup>8</sup>. This fine-tuning improves the model’s ability to generate speech in German and enhances code-switching between English and (Swiss-)German. Multiple checkpoints of the fine-tuned models were tested, revealing a tradeoff: minimal fine-tuning enhances code-switching, but excessive fine-tuning causes the output to sound overly German. Finding the right balance not only improves code-switching but also makes the generated audio sound more like an L2 speaker, incorporating German influences. This approach is particularly suitable for language learning and multilingual applications. While fine-tuning showed promise, this study ultimately used the original model checkpoint without fine-tuning to focus on other aspects, such as error preservation.

To control code-switching with **Coqui TTS**, the multilingual model “xtts\_v2” was tested for its ability to handle multiple languages. Tests with English text containing German words showed that the model struggles with automatic language switching. To address this, language tags like “[de]” for German and “[en]” for English were used to explicitly specify the language. Two approaches were tested: tagging the entire sentence (e.g., “[DE]The ball is in the Garten. Go and get it.”) and tagging at the code-switching points (e.g., “The ball is in the [de]Garten. [en]Go and get it.”). The latter approach showed promising results, but further tests are necessary to fully understand and refine language control with Coqui TTS.

8: <https://huggingface.co/marduk-ra/F5-TTS-German>

### 6.2.2 Gradio App for Evaluation

A Gradio app was developed for human evaluation, offering remote access through its sharing functionality while ensuring

9: Gradio servers act only as proxies and do not store data:  
<https://www.gradio.app/guides/sharing-your-app>

data privacy<sup>9</sup>. All information remains on the local machine where the app is executed. The app must be launched first and can then be shared via a URL. Each evaluation generates a unique, random session ID, and no additional evaluator-related information is stored. The app is designed to be self-explanatory, providing users with clear instructions before guiding them through the evaluation process. Screenshots of the instruction page and the evaluation interface are included in the Appendix B in Figure B.1.

The app lets evaluators compare synthesized speech with reference audio, rating speaker similarity and naturalness. Audio samples are presented anonymously and in random order, with evaluators hearing both the original and generated audio along with the text used for synthesis. Evaluators assign SMOS scores (1 to 5, in 0.5 intervals) and CMOS scores (-3 to +3, in steps of 1) using a simple interface with clear instructions. After submitting scores for a system, they move to the next, without the option to revisit previous samples.

Using Gradio with WandB logging has proven highly effective. The initial setup is manageable, and the benefits are substantial. Gradio's low-code interface allows for easy creation of interactive apps, enabling unlimited remote evaluators without the need for personal instruction, even supporting simultaneous sessions. Results are automatically logged and can be analyzed after all sessions. This approach is flexible and easily adaptable to other use cases.

### 6.2.3 SMOS and CMOS Evaluation Results

Table 6.1 presents the evaluation results for three TTS systems based on SMOS and CMOS scores. Among the evaluated systems, F5 TTS achieved the highest scores with a SMOS of  $3.10 \pm 1.10$  and a CMOS of  $-0.79 \pm 1.37$ , making it the best-performing system in this evaluation.

**Table 6.1:** Evaluation results for different TTS systems, showing SMOS and CMOS values as mean  $\pm$  std.

System	SMOS	CMOS
Coqui TTS	$2.71 \pm 1.09$	$-0.92 \pm 1.44$
E2 TTS	$3.01 \pm 1.04$	$-1.06 \pm 1.26$
F5 TTS	$3.10 \pm 1.10$	$-0.79 \pm 1.37$

While F5 TTS achieved the highest scores, there is still room for improvement. An SMOS of 3.10 suggests moderate speaker similarity, meaning the synthetic voices resemble the target but are not highly convincing. A CMOS of -0.79 indicates that while F5 is preferred over the other systems, its naturalness is still below the reference audio. These results suggest that improvements in prosody, articulation, and expressiveness could enhance both speaker similarity and naturalness.

For more details about scoring Figure B.2 in Appendix B illustrates the relationship between the scores for the three TTS systems. Each

plot highlights the distribution of scores, where point size and color represent the frequency of occurrences. Differences in point distributions reveal varying system performances. Most systems cluster around CMOS values of -1 to 0 and SMOS values of 2.5 to 3, indicating moderate subjective quality and slight preference for the reference. However, F5 stands out with more SMOS scores around 3, suggesting it offers more consistent or slightly better quality.

#### 6.2.4 DataSpeech Evaluation Results

The linear regression results in Table 6.2 quantify the alignment between voice reference audio annotations and synthetic audio annotations across the three TTS systems. F5 consistently outperforms the other models, achieving the highest  $R^2$  values across most features, particularly for utterance pitch mean ( $R^2 = 0.414$ ), STOI ( $R^2 = 0.239$ ), and PESQ ( $R^2 = 0.235$ ), indicating stronger correlations between original and generated annotations. E2 shows moderate performance, with lower correlations but more stable slopes, suggesting systematic deviations rather than random variation. Coqui exhibits the weakest alignment, with significantly lower  $R^2$  values, particularly for SI-SDR ( $R^2 = 0.022$ ) and PESQ ( $R^2 = 0.030$ ), indicating higher discrepancies between original and synthesized speech.

Overall, these results highlight F5 as the most reliable system for preserving acoustic features, while E2 and Coqui show greater deviations that may affect perceptual similarity and intelligibility. These trends are further visualized in Figure B.3 in Appendix B, which presents scatter plots and fitted regression lines for the alignment between DataSpeech annotations of reference and synthetic audio. This evaluation therefore confirms the subjective evaluation.

**Table 6.2:** Compact Linear Regression Results for TTS Systems (Best values in **bold**). Quantifies the alignment between voice reference and synthetic audio annotations across three TTS models. F5 demonstrates the highest correlation ( $R^2$ ) for key acoustic features, indicating superior preservation of speech characteristics.

Feature	F5			E2			Coqui		
	$R^2$	Slope	Intercept	$R^2$	Slope	Intercept	$R^2$	Slope	Intercept
Pitch Mean	<b>0.414</b>	0.660	66.84	0.380	0.588	90.73	0.191	0.413	94.07
Pitch Std	<b>0.216</b>	0.517	26.42	0.190	0.405	21.17	0.142	0.361	41.83
SNR	<b>0.215</b>	0.609	21.87	0.157	0.476	28.31	0.064	0.478	26.48
C50	0.380	0.711	9.77	<b>0.425</b>	0.693	7.31	0.011	0.128	42.95
STOI	<b>0.239</b>	0.488	0.48	0.117	0.357	0.58	0.084	0.150	0.83
SI-SDR	<b>0.157</b>	0.467	8.91	0.045	0.258	10.61	0.022	0.123	17.48
PESQ	<b>0.235</b>	0.711	0.97	0.094	0.444	1.38	0.030	0.239	2.28

### 6.2.5 Synthetic Audio Dataset

The final dataset contains **51,772 samples** totaling **102.27 hours** of synthesized speech, with a maximum length of 12 seconds and text preprocessing consistent with the ChaLL dataset. The audio samples have been generated using F5 TTS. Since the audio was generated using real children's voice references, the dataset cannot be publicly released.

For synthesis, the default inference settings of F5-TTS were used, ensuring high-quality and stable speech generation. The model operates at a 24 kHz target sample rate with 100 mel channels, a hop length of 256, and a Fast Fourier Transform (FFT) window size of 1024, using Vocos as the mel spectrogram vocoder. Key synthesis parameters include a target RMS of 0.1, cross-fade duration of 0.15, and a CFG strength of 2.0, with ODE-based inference (Euler method) and 32 function evaluations per step (NFE 32). The sway sampling coefficient (-1.0) and default speed (1.0x) ensure smooth and natural prosody, while fix\_duration is left undefined, allowing for adaptive timing adjustments. Future work could explore optimizing these inference settings to further enhance speech quality, particularly in terms of speaker similarity and naturalness.

## 6.3 Conclusion and Future Work

The effectiveness of training on augmented data depends significantly on the quality of the generated audio. As a result, selecting an appropriate TTS system and implementing it to create a synthetic dataset is important. To address the second research objective, an assessment of TTS systems was conducted using subjective metrics, where nine human evaluators compared real audio samples with their synthetic counterparts. The evaluation identified the F5 TTS system as the best performer, achieving the highest scores for both similarity (SMOS,  $3.10 \pm 1.10$ ) and naturalness (CMOS  $0.79 \pm 1.37$ ).

However, the high variance in the scores indicates variability in the evaluators' perceptions of similarity and naturalness. This suggests that while F5 TTS performed better overall, individual preferences or inconsistencies in judgment may have influenced the results. Such variability highlights the subjective nature of the evaluation and emphasizes the importance of complementing these findings with objective metrics to provide a more comprehensive assessment of the TTS system's performance. To account for variations in annotator reliability and improve the robustness of the evaluation results, using a tool like MACE (Multi-Annotator Competence Estimation) would be beneficial. MACE can help identify and adjust for inconsistencies in annotator judgments, leading to more reliable and accurate assessments.

To validate the subjective assessment, 400 additional synthesized text samples were annotated alongside their reference voices using DataSpeech. Linear regression analysis revealed that F5 outperformed E2 and Coqui, achieving the highest correlation across most features, particularly in SNR, STOI, and PESQ, indicating best preservation of intelligibility and speech quality.

With the selection of the F5 TTS system for speech data generation, the next research question focuses on how to use it to generate a synthetic speech corpus. This involves the definition of suitable reference speech samples based on duration (4-12 seconds) and minimum word count (4). These criteria ensure that each sample contains sufficient information. Subsequently, each generated text sample from Chapter 5 was randomly paired with a reference speaker. The TTS system then synthesized speech for each pair, producing the synthetic speech dataset.

This approach provides flexibility in generating speech for any text prompt using any voice reference, effectively addressing the third research question by enabling the synthesis of diverse speech samples. However, this flexibility increases the risk of poor-quality samples. To further improve and ensure the quality of the generated audio, advancements are needed in both the selection of voice references and the filtering of the synthesized output. TTS audio data can be filtered using methods such as speaker embedding similarity (as described by (W. Wang et al., 2021)) or CER or WER filtering, by back-translating the generated audio.

Another potential improvement involves adapting a pretrained adult speech TTS model to targeted child-specific data. This approach could leverage transfer learning techniques to fine-tune the model, ensuring it effectively captures the unique acoustic and prosodic characteristics of children's speech, thereby improving the naturalness and quality of the synthesized audio. This approach could also incorporate child-specific speech characteristics, such as code-switching and disfluencies. To address code-switching, preliminary experiments have been conducted using a fine-tuned variant of the F5 TTS system. The results indicate that incorporating additional German training data enhances the system's ability to handle code-switching effectively, demonstrating the potential for improving multilingual speech synthesis through targeted fine-tuning.

# 7

# Train ASR with Real and Synthetic Data

7.1 Methods . . . . .	62
7.2 Results and Deliverables . . . . .	68
7.3 Conclusion and Future Work . . . . .	70

This chapter addresses SRQ4, examining the extent to which synthetic speech data, when combined with real datasets, enhances ASR performance. The focus is on improving transcription accuracy and preserving common errors made by second-language learners. To explore this, Facebook’s `wav2vec2-xls-r-300m` model is fine-tuned using both real and synthetic speech. The real data is sourced from the ChaLL dataset (Michot et al., 2024), which contains English speech from Swiss primary school children. The synthetic data, generated as described in Chapter 6, includes controlled ERRANT error types.

## 7.1 Methods

By optimizing the model with a linear projection for character prediction, incorporating English characters, German umlauts, and a token for non-English characters, this study evaluates transcription performance without language model decoding. The goal is to determine how synthetic speech contributes to robust ASR systems for second-language learners. The impact of synthetic speech data is assessed by fine-tuning Facebook’s `wav2vec2-xls-r-300m` model with varying proportions of real and synthetic data (Section 7.1.2). Data preparation, including partitioning and concatenation, is detailed in Section 7.1.1. Evaluation focuses on overall transcription accuracy in children’s speech and the preservation of learner errors. In addition to overall WEPR, error preservation analysis examines the preservation of specific ERRANT error types (Section 7.1.3).

The `wav2vec2-xls-r-300m` model was selected for its efficiency and suitability for fine-tuning on limited-domain data. Unlike Whisper, which integrates a strong language model, `wav2vec2-xls-r` enables a more controlled evaluation of transcription and error preservation without external priors (Michot et al., 2024). The 1B model, while more powerful, was not chosen due to its significantly higher computational demands, making it less practical for this study’s fine-tuning approach.

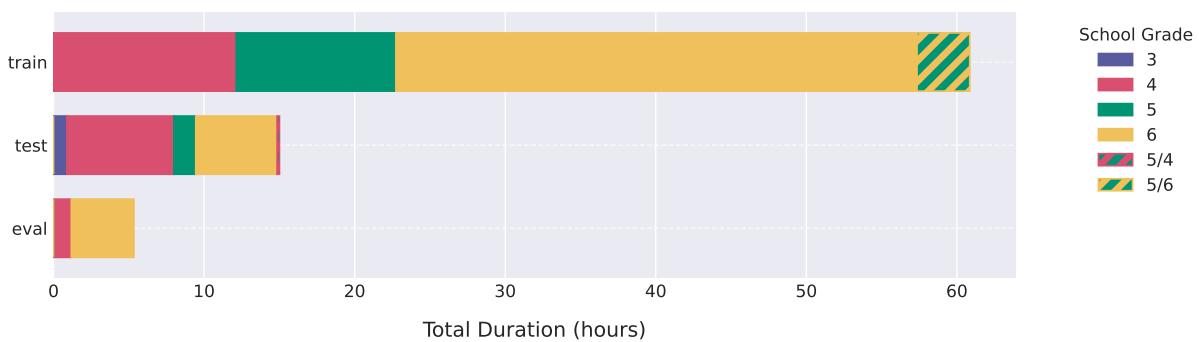
### 7.1.1 Data Preparation and Partitioning

1: <https://huggingface.co/datasets/mict-zhaw/chall/blob/main/chall.py>

The **real** data is loaded from the Hugging Face dataset repository<sup>1</sup> using a newly defined custom builder configuration (`asr_mt`). This ensures consistency and facilitates reproducibility by enabling the reuse of the same configuration across experiments. It supports

splitting audio into segments based on defined maximum durations (12 seconds) and minimum durations (0.5 seconds), while also removing trailing pauses and handling long silences with a maximum pause length of 12 seconds. Text preprocessing includes case normalization (lowercase), number-to-word conversion, and filtering allowed characters to ensure consistency in the dataset. These settings are based on the previous work (Michot et al., 2024).

The implemented stratified splitting mechanism in the dataset builder divides the dataset into train, test, and evaluation splits based on intervention IDs. The ChaLL dataset was collected across 21 interventions involving different school classes. Using intervention IDs as the splitting criterion ensures that individual speakers are not represented in multiple splits, maintaining strict separation. Furthermore, the splits were selected to ensure that each subset contains different school grades, preventing potential biases from grade-specific speech patterns. The splits defined for this work are summarized in Figure 7.1.



**Figure 7.1:** Stacked bar plot showing the distribution of school grades (3–6) across different dataset splits, with durations converted to hours. The dataset is divided into three splits: **Test** (approx. 15 hours) includes interventions [5, 20, 21, 2, 6, 10]; **Evaluation** (approx. 5 hours) includes interventions [4, 18, 19]; and **Training** (approx. 60 hours) includes interventions [1, 3, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17]. Mixed-grade data is represented with hatched bars.

The allocation of durations (60h for training, 5h for evaluation, and 15h for testing) reflects the need for a robust training set while reserving sufficient data for reliable evaluation and testing. The intervention with ID 5, which is human-annotated with ERRANT error types, is in the test split.

The generated data is loaded and subjected to similar preprocessing steps as the real data to maintain consistency. This preprocessing involves filtering audio samples shorter than 0.5 seconds or longer than 12 seconds. Additionally, the generated text undergoes the same preprocessing, including case normalization (conversion to lowercase), number-to-word conversion, and filtering of allowed characters, ensuring uniformity across the dataset. To evaluate on synthetic speech data, additional samples were generated to ensure that the evaluation data was entirely separate from the synthetic training data.

Audio durations for each sample are calculated and stored as metadata for both datasets. After preprocessing and extracting relevant information, the datasets are saved locally and reloaded for training. Each experiment (e.g., 60 hours of real data and 20 hours of synthetic data) begins by loading the complete datasets. To ensure diversity, the real and synthetic datasets are individually shuffled, and samples are iteratively selected based on cumulative durations until the specified target duration is reached. This process guarantees that the training and evaluation datasets contain the required amount of diverse audio data. After selecting the required amount of audio data, datasets from different corpora are concatenated into unified “train” and “eval” splits, combining real and synthetic data for a consistent structure. A final shuffle is applied to mix synthetic and real data, reducing any order bias that could affect model training.

### 7.1.2 Training and Evaluation Strategy

The objective is to evaluate transcription accuracy and error preservation across various combinations of real and synthetic data. This is achieved by systematically comparing different proportions shown in the grid in Table 7.1, which outlines different proportions of real and synthetic data.

**Table 7.1:** Grid of real and synthetic data combinations for ASR training and evaluation. Rows represent hours of real data, columns represent hours of synthetic data.

real/synth	0	20	40	60	80
0	-	0_20	0_40	0_60	0_80
60	60_0	60_20	60_40	60_60	60_80

Although other combinations, such as 40 hours of real data paired with 20 hours of synthetic data, could also be informative, the primary focus is on utilizing all available real data paired with varying proportions of synthetic data. Furthermore, synthetic data is investigated independently, without any real data, to assess whether its distribution aligns with that of the real data.

To assess the impact of fine-tuning, each configuration is used to train the wav2vec2-xls-r-300m model. Unlike in the previous work (Michot et al., 2024), this study does not employ a cross-validation training approach due to the exponential growth in the number of required experiments. Specifically, for each fold, training nine additional models would be necessary, making cross-validation computationally impractical. Therefore, a simple split is used instead.

Training is conducted on four NVIDIA Tesla V100 GPUs, with a maximum of 150 epochs or until the evaluation loss converged. Data loading and preprocessing is performed on the first GPU, while the remaining GPUs access the preprocessed data from cache. The training procedure employed a learning rate of  $1 \times 10^{-4}$ , a per-device batch size of 17, and 19 gradient accumulation steps.

This results in an effective batch size of 1292, which corresponds to approximately two hours of audio per batch. The optimization was performed using the `adamw_hf` optimizer.

During each experiment, the WER is computed at every validation step using the evaluation set. If an improvement in WER is observed, the corresponding model checkpoint is saved as the new best model. The final evaluation is conducted using these best-performing checkpoints.

For each combination in the grid, performance is evaluated on both real and synthetic data. Real test data is assessed for transcription accuracy (WER) and error preservation (WEPR). Synthetic data is evaluated based on WER and the ERRANT error types generated during text synthesis. However, WEPR cannot be applied to the synthetic test data, as it lacks the ChaLL annotations. Both real and synthetic data are filtered, ensuring that the test set includes only samples containing a minimum of three words. Predictions during testing are generated using a beam search decoding strategy with a beam size of 100 to optimize transcription accuracy.

### 7.1.3 ERRANT Error Type Evaluation

While WEPR provides a measure of the ASR model's overall precision in preserving errors, it does not offer detailed error classification. However, in the ChaLL setting, detailed error annotations are crucial for downstream tasks. A specialized model for annotating errors in transcripts has been developed as part of the ChaLL initiative, but access to this model was not available for this work. Thus, an alternative approach is proposed, involving a two-step process. First, the transcript is corrected using a GEC model. Next, the edits between the original transcript and the corrected version are classified using ERRANT. In this study, the **T5 Grammar Correction** model<sup>2</sup> was used as a tool for GEC.

2: <https://huggingface.co/vennify/t5-base-grammar-correction>

#### 7.1.3.1 Simple Error Detection

The simplest implementation of this approach involves generating a single GEC-correction and applying edit classification between the original text and its corrected version. Table 7.2 provides examples of this method.

The baseline approach can be extended by generating multiple corrections instead of a single one. Since the focus is on identifying edits and error types rather than the corrections themselves, multiple correction sequences can help detect and combine errors. Annotations from these corrections can either be aggregated (Section 7.1.3.2) or selected using a majority vote (Section 7.1.3.3).

### 7.1.3.2 Multiple Corrections Union

Using multiple corrections results in an equal number of ERRANT alignments, each specifying its own edits and classifying errors independently. However, these alignments can be merged. To unify alignments, edits (i.e., error positions) are first consolidated, and the most frequently occurring error type is selected for each edit. A key challenge in this approach is overlapping spans. While most annotations involve single-token errors, certain error types, such as word order errors, span multiple tokens. Currently, overlapping spans are accepted, but future refinements may be necessary. Table 7.3 provides examples of this method.

### 7.1.3.3 Multiple Corrections Majority

Instead of unifying edits and selecting error types per edit, the errors can be determined by **error majority vote**. In this method, an error (considering both its position and error type) is considered detected only if it appears in at least half of the generated outputs. Table 7.4 shows examples using the error majority vote.

As an alternative to the error majority vote, a **two-step majority voting** process can be applied to determine error positions and

**Table 7.2:** Error Detection Examples for **Simple Error Detection**. Errors are identified by comparing original text to a correction using the ERRANT annotation framework.

Generated Error Type, Text, Correction	Detected Error Type
Error: R:VERB:INF Orig.: I <b>buyed</b> a new chat toy yesterday. It is very cute. Corr.: I <b>bought</b> a new chat toy yesterday. It is very cute.	((1, 2), R:VERB:INF)
Error: R:WO Orig.: I <b>on</b> put my boots before going outside. Corr.: I put my boots before going outside.	((1, 2), U:ADV)
Error: R:SPELL, R:VERB:SVA Orig.: I <b>recomend</b> a coat in November because we <b>has</b> cold weather. Corr.: I <b>recommend</b> a coat in November because we <b>have</b> cold weather.	((1, 2), R:SPELL) ((8, 9), R:VERB:SVA)

**Table 7.3:** Error Detection Examples for **Multiple Corrections Union**. Error candidates are identified by comparing original text to multiple GEC-corrections using the ERRANT annotation framework. Here, errors are determined by unifying all error positions and selecting the most frequently occurring error per position.

Generated Error Type, Text, Correction	Detected Error Type
Error: R:VERB:INF Orig.: I <b>buyed</b> a new chat toy yesterday. It is very cute. 1 Corr.: I <b>bought</b> a new chat toy yesterday. It is very cute. 2 Corr.: I <b>bought</b> a new chat toy yesterday <b>and</b> it is very cute. 3 Corr.: I <b>bought</b> a new chat toy yesterday, it is very cute.	((1, 2), R:VERB:INF) ((7, 9), R:OTHER)
Error: R:WO Orig.: I <b>on</b> put my boots before going outside. 1 Corr.: I put my boots <b>on</b> before going outside. 2 Corr.: I put my boots before going outside. 3 Corr.: I on put my boots before going outside.	((1, 2), U:ADV) ((5, 5), M:PART)
Error: R:SPELL, R:VERB:SVA Orig.: I <b>recomend</b> a coat in November because we <b>has</b> cold weather. 1 Corr.: I <b>recommend</b> a coat in November because we <b>have</b> cold weather. 2 Corr.: I <b>recommend wearing</b> a coat in November because we <b>have</b> cold weather. 3 Corr.: I <b>recomend</b> a coat in November because we <b>have</b> cold weather.	((1, 2), R:SPELL) ((8, 9), R:VERB:SVA)

types more robustly. First, majority voting is used to identify the most frequently annotated error positions in the original string. Only positions that appear in more than half of the corrections are retained. Then, for each retained position, the most frequently assigned error type is selected. This approach ensures that both error localization and classification are based on the most consistent annotations across multiple corrections. Table 7.5 shows examples using the two-step majority vote.

**Table 7.4:** Error Detection Examples for **Multiple Corrections Error Majority**. Errors are identified by comparing original text to multiple GEC-corrections using the ERRANT annotation framework. Here, errors are detected through a majority vote.

Generated Error Type, Text, Correction	Detected Error Type
Error: R:VERB:INF Orig.: I <b>buyed</b> a new chat toy yesterday. It is very cute. 1 Corr.: I <b>bought</b> a new chat toy yesterday. It is very cute. 2 Corr.: I <b>bought</b> a new chat toy yesterday <b>and</b> it is very cute. 3 Corr.: I <b>bought</b> a new chat toy yesterday, it is very cute.	((1, 2), R:VERB:INF) ((7, 9), R:OTHER)
Error: R:WO Orig.: I <b>on</b> put my boots before going outside. 1 Corr.: I put my boots <b>on</b> before going outside. 2 Corr.: I put my boots before going outside. 3 Corr.: I <b>on</b> put my boots before going outside.	((1, 2), U:ADV) ((5, 5), M:PART)
Error: R:SPELL, R:VERB:SVA Orig.: I <b>recomend</b> a coat in November because we <b>has</b> cold weather. 1 Corr.: I <b>recommend</b> a coat in November because we <b>have</b> cold weather. 2 Corr.: I <b>recommend wearing</b> a coat in November because we <b>have</b> cold weather. 3 Corr.: I <b>recomend</b> a coat in November because we <b>have</b> cold weather.	((1, 2), R:SPELL) ((8, 9), R:VERB:SVA)

**Table 7.5:** Error Detection samples for **Multiple Corrections Span Majority**: Examples of ERRANT error detection applied to generated samples, where errors are identified by comparing original to multiple GEC-corrections using the ERRANT annotation framework. Here, errors are detected through a two-step majority vote: first on the position then on the error type.

Generated Error Type, Text, Correction	Detected Error Type
Error: R:VERB:INF Orig.: I <b>buyed</b> a new chat toy yesterday. It is very cute. 1 Corr.: I <b>bought</b> a new chat toy yesterday. It is very cute. 2 Corr.: I <b>bought</b> a new chat toy yesterday <b>and</b> it is very cute. 3 Corr.: I <b>bought</b> a new chat toy yesterday, it is very cute.	((1, 2), R:VERB:INF) ((7, 9), R:OTHER)
Error: R:WO Orig.: I <b>on</b> put my boots before going outside. 1 Corr.: I put my boots <b>on</b> before going outside. 2 Corr.: I put my boots before going outside. 3 Corr.: I <b>on</b> put my boots before going outside.	((1, 2), U:ADV) ((5, 5), M:PART)
Error: R:SPELL, R:VERB:SVA Orig.: I <b>recomend</b> a coat in November because we <b>has</b> cold weather. 1 Corr.: I <b>recommend</b> a coat in November because we <b>have</b> cold weather. 2 Corr.: I <b>recommend wearing</b> a coat in November because we <b>have</b> cold weather. 3 Corr.: I <b>recomend</b> a coat in November because we <b>have</b> cold weather.	((1, 2), R:SPELL) ((8, 9), R:VERB:SVA)

#### 7.1.3.4 Multiple Corrections Probabilistic

Ultimately, the approach can be finalized by also considering the probabilities of each corrected sequence. For each edit position, error types are annotated and their cumulative probabilities aggregated across all corrections. The most probable error type is selected for each edit, ensuring alignment with the likelihoods provided by the corrections. Additionally, spans with a cumulative

probability below a specified threshold (the highest sequence probability) are filtered out, ensuring that only reliable and significant errors are retained.

**Table 7.6:** Error Detection samples for **Multiple Corrections Probabilistic**: Examples of ERRANT error detection applied to generated samples, where errors are identified by comparing original to multiple GEC-corrections using the ERRANT annotation framework. Here, errors are detected through a majority vote on both edits and corresponding error types based on probabilities.

Generated Error Type, Text, Correction (Probability)	Detected Error Type
Error: R:VERB:INF Orig.: I <b>buyed</b> a new chat toy yesterday. It is very cute. 1 Corr. (0.923): I <b>bought</b> a new chat toy yesterday. It is very cute. 2 Corr. (0.855): I <b>bought</b> a new chat toy yesterday, it is very cute. 3 Corr. (0.818): I <b>bought</b> a new chat toy yesterday <b>and</b> it is very cute.	((1, 2), 'R:VERB:INFL', 2.598) ((7, 9), 'R:PUNCT', 0.854)
Error: R:WO Orig.: I <b>on</b> put my boots before going outside. 1 Corr. (0.880): I put my boots before going outside. 2 Corr. (0.862): I on put my boots before going outside. 3 Corr. (0.793): I put my boots <b>on</b> before going outside.	((1, 2), 'U:ADV', 1.672) ((5, 5), 'M:PART', 0.793)
Error: R:SPELL, R:VERB:SVA Orig.: I <b>recomend</b> a coat in November because we <b>has</b> cold weather. 1 Corr. (0.953): I <b>recommend</b> a coat in November because we <b>have</b> cold weather. 2 Corr. (0.785): I <b>recommend wearing</b> a coat in November because we <b>have</b> cold weather. 3 Corr. (0.777): I <b>recomend</b> a coat in November because we <b>have</b> cold weather.	((1, 2), R:SPELL, 0.953) ((8, 9), R:VERB:SVA, 2.515)

## 7.2 Results and Deliverables

This section presents the outcomes of this chapter, divided into evaluation on real and synthetic data. The evaluation on real data (Section 7.2.1) reports performance in terms of transcription accuracy (WER) and error preservation (WEPR) on real test data. The evaluation on synthetic data (Section 7.2.2) examines performance of transcription accuracy (WER) and ERRANT error preservation on synth test data.

### 7.2.1 Evaluation on Real Data

The evaluation of ASR models on real data yielded transcription accuracy results, presented in Table 7.7, and error preservation scores, detailed in Table 7.8.

**Table 7.7:** WER scores for different combinations of real and synthetic training data, evaluated on real test data. Relative differences in the 60h row are computed relative to 60\_0 (0.229).

Real \ Synth	0h	20h	40h	60h	80h
0h	–	0.699	0.624	0.630	0.616
60h	0.229	0.227	0.228	0.233	<b>0.226</b> (0.0%) (-0.75%) (-0.52%) (+1.67%) (-1.19%)

**Table 7.8:** WEPR scores for different combinations of real and synthetic training data, evaluated on real test data. Relative differences in the 60h row are computed relative to 60\_0 (0.278).

Real \ Synth	0h	20h	40h	60h	80h
0h	–	0.648	0.557	0.566	0.587
60h	0.278	<b>0.269</b> (0.0%) (-3.17%)	0.272 (-2.17%)	0.275 (-1.06%)	0.271 (-2.73%)

The results show that increasing synthetic data generally improves transcription accuracy slightly, as indicated by WER scores. The lowest WER (0.226) is achieved with 60h real and 80h synthetic data, though 60h real with 60h synthetic disrupts the trend of more data consistently improving results. A similar pattern is observed for WEPR, where all 60h real data configurations improve error preservation, with the lowest value (0.269) at 60h real and 20h synthetic data. Compared to the baseline without synthetic data, additional 80h of synthetic data improves accuracy by 1.19% and error preservation by 2.73%.

Models trained solely on synthetic data exhibit high WER and WEPR, indicating a distribution mismatch with real data. Nevertheless, increasing synthetic data improves both accuracy and error preservation.

### 7.2.2 Evaluation on Synthetic Data

The accuracy results of testing the ASR models on synthetic data are summarized in Table 7.9. The table presents the WER for each combination of real and synthetic training data, evaluated on synthetic test data.

Real\Synth	0	20	40	60	80
0	–	0.144	0.113	0.092	0.079
60	0.309	0.109	0.093	0.084	0.071
	(–)	(-64.72%)	(-69.90%)	(-72.82%)	(-77.02%)

**Table 7.9:** WER scores for different combinations of real and synthetic training data, evaluated on synthetic test data. Relative differences in the 60 row are computed relative to 60\_0 (0.309).

The results align with expectations based on synthetic data experiments. Models trained with synthetic data outperform those trained exclusively on real data. The model trained solely on real data (60\_0) exhibits a WER of 0.309, significantly higher than any configuration incorporating synthetic data. Adding 20 hours of synthetic data (60\_20) reduces WER by 64.72% relative to the baseline. Moreover, models trained with synthetic data show a consistent decrease in WER as the amount of synthetic training data increases.

The results indicate that the distribution of real and synthetic data differ. If the distributions were the same, a model trained solely on real data would be expected to perform similarly to those trained on synthetic data when evaluated on synthetic test data. However, the significantly higher WER of the real-only model (60\_0) compared to models trained with synthetic data suggests a distributional mismatch between real and synthetic data.

**ERRANT Error Evaluation** Applying the different error detection strategies described in Section 7.1.3 resulted in the scores summarized in Table C.2 in Appendix C. Each row represents the evaluation of an ASR model using a specific error detection

strategy. The scores are obtained by comparing the GEC-corrected, ERRANT-annotated transcript prediction with:

- ▶ “**Errors**” – The error types used as input to generate a synthetic text sample (as described in Chapter 5).
- ▶ “**Spans**” – The actual ERRANT-detected errors identified by comparing the grammatically correct and incorrect versions of generated text samples.

The scores are calculated the same way as described in Section 5.2.4.3. Because scores in the “Errors”-column also account for the inaccuracies during text samples generation (with a precision of 0.608 as described in Section 5.2.4.3), the scores are lower compared to “Spans”.

Given the multiple steps involved in generating the synthetic data for this analysis, numerous potential sources of inaccuracies exist throughout the pipeline. As a result, the scores remain relatively low. The results show that `multi_union` consistently achieves the highest F1 scores across both “Errors” and “Spans”. Overall, increasing synthetic training data improves scores, but the trend is not strictly linear, with some fluctuations in precision and recall. Among the strategies, `multi_prob` and `multi_span_majority` also perform well, striking a balance between precision and recall, while `simple` generally yields lower scores, particularly in recall. Some models can retain up to 30% of the errors in a synthesized sample, meaning that these errors persist through both the TTS and TTS processes.

Furthermore, these results are not directly comparable to human-annotated errors in real data. A similar experiment was conducted on the human-annotated subset; however, the results were highly imprecise. For completeness, this evaluation is included in this work, but its findings lack statistical significance. The primary takeaway is that the automatic error detection approach using GEC and ERRANT on real-world data is not viable. The corresponding results are presented in Table C.3 in Appendix C.

### 7.3 Conclusion and Future Work

This chapter investigated the impact of synthetic speech data on ASR training, assessing both transcription accuracy and the preservation of learner errors. The results indicate that while synthetic data contributes to performance improvements, the relationship between data composition and model effectiveness is complex and not strictly linear. The best-performing model, trained with 60 hours of real and 80 hours of synthetic data (**60\_80**), achieved the lowest WER (0.226) and a WEPR (0.271), representing a 1.19% improvement in transcription accuracy and a 2.73% improvement in error preservation compared to the baseline (**60\_0**). However, the significant performance gap between real-only and synthetic-trained

models when tested on synthetic data suggests a distributional difference between real and synthetic data.

These results demonstrate the impact of synthetic data on ASR performance, but fluctuations suggest that data partitioning influences effectiveness. To obtain more reliable insights, the grid could be expanded to include all possible combinations of real and synthetic data in 20-hour increments, potentially revealing clearer trends. If inconsistencies persist, a k-fold cross-validation strategy may be necessary. This would mitigate the effects of data variability, ensuring that results are not biased by a specific partition and providing a more robust evaluation of training configurations.

The evaluation of ERRANT error types on synthetic data assessed the pipeline’s ability to preserve errors from generation to detection. While this is an important aspect of this work and the idea of optimizing an ASR system capabilities in preserving certain types of errors, this approach in its current form is inadequate. It is influenced by multiple steps, making the results hard to interpret. While adding synthetic data to the training led to modest improvements, this approach does not reliably measure overall error type preservation. Key limitations include potential errors in ASR transcription, GEC correction, and ERRANT classification. Without language model decoding, ASR models may introduce artifacts that the GEC model attempts to correct, further complicating error detection. Additionally, ERRANT struggles with large discrepancies between original and corrected transcripts. Addressing these challenges is essential for more reliable error assessment. In this form, no definitive conclusions can be drawn regarding the preservation of specific errors.

# 8

# Conclusion & Future Work

8.1 Conclusion . . . . .	72
8.2 Future Work . . . . .	74

The increasing reliance on ASR systems across various domains underscores the importance of developing models that can effectively handle diverse speaker profiles, including second-language learners and children. However, ASR systems often struggle with the spontaneous and error-prone speech of language learners, where unique phonetic, grammatical, and semantic deviations from native speaker norms are prevalent. This challenge is particularly pronounced in the context of children's speech, where additional variances in pronunciation, intonation, and spontaneity further complicate transcription accuracy.

This research investigates the potential of synthetic speech data to address these challenges. Specifically, it examines whether incorporating controlled and diverse synthetic speech samples can improve ASR performance in transcribing second-language learners' speech. By focusing on speech patterns derived from the ChaLL corpora, the study aims to explore how accurately synthetic data can replicate learners' speech characteristics, including common errors, and whether these samples can enhance transcription accuracy and preserve error patterns. This final chapter summarizes key findings and implications (Section 8.1), followed by potential improvements and future research directions for enhancing ASR systems in second-language acquisition (Section 8.2).

## 8.1 Conclusion

This study investigated the potential of synthetic speech data to enhance ASR performance for second-language learners' speech. Through controlled generation of text samples and zero-shot synthesis of children's voices, it explored how synthetic corpora can replicate spontaneous and error-prone speech patterns observed in the ChaLL corpora. The ultimate research objective was to fine-tune ASR models by incorporating varying proportions of real and synthetic data to assess the impact on transcription accuracy (WER), and the preservation of learner errors (WEPR).

The first sub-research question (SRQ1) examined how text samples can be generated to replicate spontaneous speech patterns and learner-specific errors observed in the ChaLL corpus. This study developed a multi-step sample generation framework using LLMs, producing 34,000 parallel text pairs with controlled grammatical errors. The methodology incorporated modular prompt chaining to introduce specific error types while maintaining linguistic realism. Evaluation against ERRANT annotations revealed both successes

and challenges, particularly in error classification accuracy. While the generated dataset aligns with real learner errors to a degree, inconsistencies in detection highlight areas for refinement. Future improvements should focus on optimizing error selection, refining prompts, and incorporating alternative annotation frameworks to enhance the reliability of synthetic text samples. To fully answer the question of accurately replicating learner speech patterns, additional evaluation strategies, such as human assessment or alternative error detection models, need to be implemented.

The second and third sub-research questions (SRQ2 and SRQ3) examined the selection of an optimal TTS system for synthesizing children's voices and the generation of a 100-hour synthetic speech corpus. A comparative evaluation of three zero-shot TTS systems, Coqui, E2, and F5, was conducted using subjective human ratings for speaker similarity and naturalness, along with automated DataSpeech analysis. F5 TTS outperformed the other systems, achieving the highest scores, making it the most suitable choice for voice cloning. Using F5, a dataset of 51,772 synthetic speech samples was generated by pairing synthetic text samples with real children's voice references. While the framework effectively synthesized speech, challenges in speaker similarity and naturalness indicate the need for refining voice reference selection and filtering synthetic speech to ensure speech quality. To improve voice reference selection, a random approach could be replaced with human selection. Effective filtering strategies, such as speaker embedding similarity, linguistic consistency checks, or phonetic alignment metrics, have been explored in previous research and should be considered in future work. Other future improvements may include hyperparameter tuning to optimize synthesis quality and model stability, as well as fine-tuning TTS models on child-specific speech to better capture the unique prosodic and phonetic characteristics of young learners.

The final research question (SRQ4) investigates how integrating different proportions of synthetic speech data with real data impacts transcription accuracy and error preservation when fine-tuning ASR models. The results indicate that combining real and synthetic data can enhance ASR performance, but the improvements are not strictly proportional to the amount of synthetic data added. The best-performing model, trained with 60 hours of real and 80 hours of synthetic speech, achieved a 1.19% reduction in WER and a 2.73% improvement in error preservation (WEPR) compared to the baseline using only real data. However, certain configurations with lower amounts of synthetic data performed comparably, suggesting that the optimal balance between real and synthetic data is not fully trivial. Models trained exclusively on synthetic speech showed significantly higher WER and WEPR, indicating a distribution mismatch with real learner speech. This is further evidenced by testing on synthetic data. While models trained with synthetic data perform well, models trained exclusively on real data achieve

significantly higher WER. Additionally, an evaluation of ERRANT error types confirmed that while synthetic data contributes to error retention, the overall preservation accuracy remains limited. These findings highlight the need for refined data selection strategies and training methodologies to maximize the benefits of synthetic speech augmentation in ASR fine-tuning.

The research question was addressed by empirically demonstrating the positive impact of synthetic data on ASR performance, showing measurable improvements in transcription accuracy and error preservation. However, further investigation is necessary to refine methodologies and explore additional optimizations. This study introduces a highly modular and customizable pipeline that serves as a foundation for future work in improving speech synthesis for ASR training. The results highlight the pipeline's potential and its adaptability for further refinement. Key challenges, including speaker similarity, linguistic realism, and accurate error preservation, require continued optimization in data selection, filtering, and model fine-tuning. Future research should build upon this framework to enhance ASR accuracy and error preservation for second-language learners.

## 8.2 Future Work

While the results demonstrate the potential of synthetic data, several challenges remain. The following sections outline key areas for future research to refine and extend the proposed approach, ultimately advancing ASR systems for language learning applications.

**Ethical and Educational Implications:** As synthetic speech becomes more prevalent in education and research, further exploration is needed to address ethical concerns, such as data privacy, and its potential impact on language learning outcomes.

**Filtering Synthetic Text Samples:** While controlled text sample generation shows promise in producing parallel sentences with predefined ERRANT error types, challenges persist in accurately generating and evaluating grammatical errors. ERRANT, as a rule-based tool, performs well in extracting edits and classifying errors. However, its accuracy is often compromised when applied to second-language speech, which contains a high density of diverse and complex errors. This limitation leads to frequent misclassifications and inconsistent results.

To address these challenges, future work should consider two critical improvements. First, exploring advanced approaches, potentially neural-based methods, could enhance the precision of error detection and classification. Such methods may offer more robust and context-aware capabilities compared to rule-based tools like ERRANT. Second, the scope of errors generated and evaluated

must be carefully narrowed. While this study aimed to encompass most of the 55 ERRANT error types, this broad scope may be impractical due to the diversity and variability of errors. Future research could focus on a smaller subset of critical error types that are more reliably generated and evaluated. This targeted approach could improve both the quality and efficiency of synthetic text sample filtering.

**Filtering TTS Data and/or Finetuning TTS System:** The two-step approach of generating text samples and subsequently using these as prompts to synthesize speech with children's audio as a reference has demonstrated considerable flexibility. However, this study highlights the need for more rigorous evaluation and filtering of reference speaker and TTS data in future work. In the current study, the only filtering applied to the synthetic data was based on audio length, which may have contributed to the limited improvements in ASR performance when incorporating synthetic speech.

To address this, future research should implement advanced filtering techniques to ensure higher quality synthetic data. Potential strategies include:

- ▶ *Automated Filtering:* Utilize, for example, speaker embeddings to measure speaker similarity between synthetic and reference audio. This approach could ensure that the synthesized speech closely matches the intended characteristics of the reference voice.
- ▶ *DataSpeech Filtering:* Use tools like DataSpeech to compare annotations of reference and generated audio, ensuring alignment between reference audio and TTS data.
- ▶ *Human Filtering:* Incorporate manual filtering and evaluation to assess the naturalness and similarity of generated audio, as automated methods alone may not fully capture subjective quality aspects.

By applying or combining these approaches, future research can enhance the quality of synthetic TTS data, potentially leading to more improvements in ASR performance. Additionally, further refinement or fine-tuning of the TTS system itself may be necessary to better capture the specific characteristics of second language children's speech.

**TTS with Code Switching:** An important yet underexplored aspect in this work is the integration of code-switching in second-language learning. Since beginners often incorporate elements of their first language into their speech, it is crucial that ASR systems accurately transcribe such utterances. Future work should focus on refining data augmentation techniques to better account for code-switching.

The proposed framework demonstrates potential in generating text with code-switched fragments, but challenges remain in audio

generation. Experiments with a fine-tuned F5 TTS model trained on German data highlighted the difficulty of achieving the right balance. Models trained minimally on German data retained an “English” sound with only a slight German accent, resulting in minimal improvements in code-switching capabilities. Conversely, models extensively trained on German data improved the pronunciation of German words but produced audio that sounded predominantly German overall, losing the desired balance between the two languages.

For Swiss language learners, improvements in TTS accuracy will require more targeted efforts. Maybe finetuning a TTS system on both Swiss-German children’s speech and the ChaLL corpus may achieve a better balance between the two languages, facilitating more accurate code-switching in synthesized speech. Addressing these aspects of synthetic data will potentially improve ASR systems for learners with Swiss-German as their first language and enhance transcription accuracy in code-switched contexts.

**Training Strategy:** The not fully linear relationship between synthetic data quantity and ASR performance suggests the need for an adapted training approach. Instead of a simple train-test-evaluation split, a k-fold cross-validation strategy could mitigate variability in both real and synthetic data. Additionally, the experimental grid was limited to configurations with either no real data or 60 hours of real data. Future research could extend this by systematically exploring all combinations of real and synthetic data in 20-hour increments to provide a more comprehensive analysis.

**Larger/Alternative Models:** Expanding model capacity is a straightforward approach to improving ASR performance. Future research could explore training larger models (e.g., Whisper) or alternative architectures to enhance transcription accuracy and error preservation. Joint training of ASR with explicit error annotations may further refine the system’s ability to capture learner-specific mistakes.

Integrating this approach with a refined data augmentation pipeline could enable ASR models to better capture child-specific speech characteristics and error patterns. If synthetic speech generation achieves greater naturalness at the child speaker level, ASR models could be trained to preserve, recognize and annotate specific types of learner errors. This would allow for the development of specialized ASR systems tailored to particular proficiency levels or pedagogical objectives. By focusing on predefined error types relevant to corrective feedback, ASR technology could be more effectively applied in language learning settings. This shift from generalized ASR models to task-specific systems would enhance their adaptability for educational applications.

**Close the Error Preservation Loop:** While this study proposed a strategy for detecting and evaluating error types in ASR transcripts, practical implementation was not feasible within the current scope

due to compounding inaccuracies across multiple steps. Transcription errors made it challenging to distinguish between learners' actual mistakes and system-generated inaccuracies. Additionally, off-the-shelf GEC tools, although effective at correcting certain issues like spelling mistakes, are not optimized for classifying learners' errors, particularly semantic errors involving missing or unnecessary words. This mismatch resulted in numerous edits between transcripts and corrections, ultimately reducing ERRANT's ability to classify errors accurately when faced with high error density. Thus, this work was not possible to fully validate the entire pipeline, from controlled error generation to error preservation in TTS synthesis and ASR transcriptions. Additional methods and tools are required to enhance this aspect.

Refining this process is crucial for accurately preserving and evaluating errors, bringing the ChaLL project closer to its goal of supporting language learning. Improving the data augmentation pipeline and optimizing TTS models would enhance the ability to assess how well TTS systems capture specific error types. Greater control over this process would enable targeted improvements in educational TTS applications. If synthetic data can be designed to replicate the acoustic characteristics of child-like speech while allowing content control, this pipeline could facilitate the training of specialized models. The feasibility and effectiveness of this approach will require further research and development.

## **APPENDIX**

# A Sample Generation

## A.1 ERRANT

**Table A.1:** There are 55 total possible error types based on Bryant et al. (2017). This table shows all of them except UNK, which indicates an uncorrected error. A dash indicates an impossible combination. The colors indicate the error codes used for generation. Error codes are color-coded for clarity: green indicates errors that are used, red indicates those that are not used, and orange marks errors that are used with caution (e.g., because they represent semantic errors).

Type	Missing	Unnecessary	Replacement
Token Tier	Adjective	M:ADJ	U:ADJ
	Adverb	M:ADV	U:ADV
	Conjunction	M:CONJ	U:CONJ
	Determiner	M:DET	U:DET
	Noun	M:NOUN	U:NOUN
	Particle	M:PART	U:PART
	Preposition	M:PREP	U:PREP
	Pronoun	M:PRON	U:PRON
	Punctuation	M:PUNCT	U:PUNCT
	Verb	M:VERB	U:VERB
Other	Contraction	M:CONTR	U:CONTR
	Morphology	-	R:MORPH
	Orthography	-	R:ORTH
	Other	M:OTHER	U:OTHER
	Spelling	-	R:SPELL
	Word Order	-	R:WO
Morphology Tier	Adjective Form	-	R:ADJ:FORM
	Noun Inflection	-	R:NOUN:INFL
	Noun Number	-	R:NOUN:NUM
	Noun Possessive	M:NOUN:POSS	U:NOUN:POSS
	Verb Form	M:VERB:FORM	U:VERB:FORM
	Verb Inflection	-	R:VERB:INFL
	Verb Agreement	-	R:VERB:SVA
	Verb Tense	M:VERB:TENSE	U:VERB:TENSE

## A.2 Sample Generation Algorithm

This algorithm describes how parallel texts consisting of a grammatically correct version and its corresponding incorrect counterpart are generated. It takes a subject, optional error creation rules, a target word count, and a flag for code-switching, then applies the specified rules to introduce grammatical errors systematically. The output is a JSON object containing both the correct and incorrect texts.

### Algorithm A.2.1

#### **Input:**

- ▶ A *subject* and its corresponding *subject type*.
- ▶ A list of *Error Creation Rules* (optional).
- ▶ A boolean flag *use\_code\_switching*.
- ▶ A *target word count*.

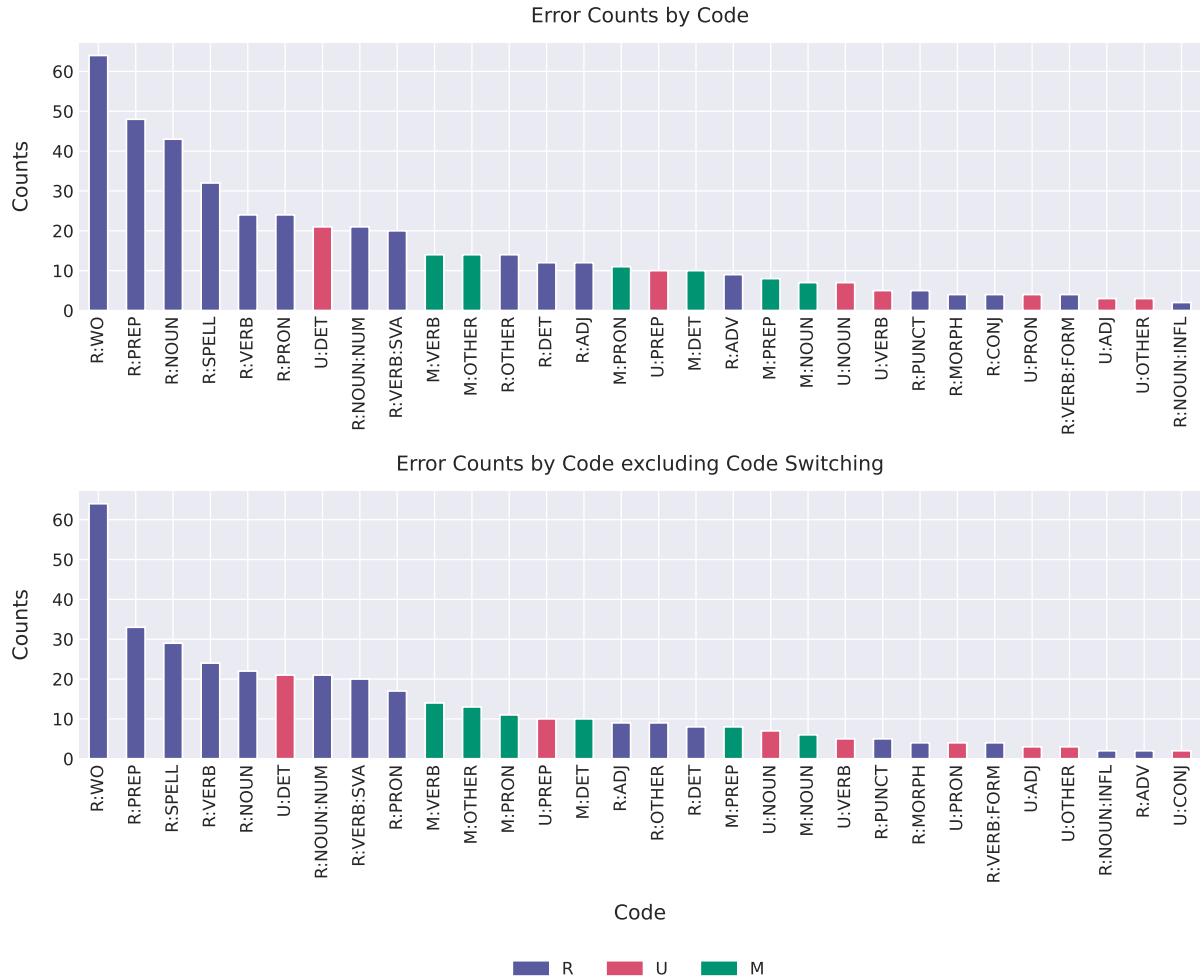
#### **Output:** A JSON object containing:

- ▶ "correct\_text": A grammatically correct text.
- ▶ "incorrect\_text": An ungrammatical version of the correct text.

#### **Prompt Generation Steps:**

1. Generate a grammatically correct text (`correct_text`) on the given subject. Ensure:
  - ▶ Sentences are short, simple, and suitable for CEFR A1-A2 learners.
  - ▶ Total word count approximates *target word count*.
2. Initialize `incorrect_text` as a copy of `correct_text`.
3. **Apply Error Creation Rules:**
  - a) If *Error Creation Rules* is empty and *use\_code\_switching* is False:
    - ▶ Set "`correct_text`" = "`incorrect_text`".
    - ▶ Exit.
  - b) For each error rule in the *Error Creation Rules*:
    - ▶ If `type_label` = "R" or `tier` ∈ {"other", "morphology"}:
      - Replace the `category_label grammar_correct` with `grammar_incorrect` in `incorrect_text`.
    - ▶ If `type_label` = "M":
      - Remove the `category_label grammar_correct` from `incorrect_text`.
    - ▶ If `type_label` = "U":
      - Introduce an unnecessary `category_label grammar_incorrect` into `incorrect_text`.
4. If `use_code_switching` = True:
  - ▶ Replace one or more words in `incorrect_text` with their German equivalents suffixed by "@g".
  - ▶ Do not append "@g" to English words.
5. Format the final `correct_text` and `incorrect_text` as a JSON object.

### A.3 Evaluation of Annotated Subset



**Figure A.1:** Comparison of ERRANT error counts by code. The top panel displays the distribution of all error types, while the bottom panel focuses on errors excluding those attributed to code-switching. The shared legend categorizes errors as R (replacement), U (unnecessary), and M (missing).

## A.4 Sample Generation Prompts

### A.4.1 Subject Generation

This chapter includes the prompts used in the sample generation framework. The background colors of the code blocks differentiate between template placeholders (blue) and filled templates with specific values (orange).

#### A.4.1.1 Subject Generation Prompt

The following code shows the general structure of the subject generation prompt, which uses placeholders to dynamically specify the number, type, and format of subjects to be generated:

```

1 Generate {subject_window_size} {subject_type} in English, suitable for
learners at the A1-A2 level of the CEFR framework.
2
3 {format_instructions}
```

#### A.4.1.2 Subject Generation Prompt Filled

This code demonstrates a filled version of the subject generation prompt with placeholders replaced by specific values, instructing the LLM to generate a list of 30 common nouns suitable for A1-A2 learners:

```

1 Generate 30 Common Noun in English, suitable for learners at the A1-A2 level
of the CEFR framework.
2
3 Your response should be a markdown list, eg: '- foo
4 - bar
5 - baz'
```

## A.4.2 Grammar Generation

This section details the prompts used for grammar generation in the sample generation framework. These prompts guide the LLM to produce structured outputs, focusing on specific grammar categories and patterns.

### A.4.2.1 Grammar Generation Prompt

The following code represents the general structure of the grammar generation prompt. It uses placeholders to dynamically specify the window size, category, and examples of grammar to be generated:

```

1 Generate a grammar list of up to {window_size} {category_label}s.
2
3 Your response should be a list of comma separated values, eg: '{example}'
```

### A.4.2.2 Grammar Pair Generation Prompt

This prompt is used to generate pairs of grammar constructs that highlight errors. It dynamically incorporates placeholders for the window size, category label, and a description of the error focus area:

```

1 Generate a list of up to {window_size} '{category_label}' pairs highlighting
grammar errors. Focus on {desc}.
2
3 Example/s: {example}
4
5 Your response should be a list of comma separated '{category_label}'
replacement error pairs, eg:
6 'wrong_1 -> correct_1, wrong_2 -> correct_2'
```

### A.4.2.3 Grammar Generation Prompt Filled

This example demonstrates a filled version of the grammar generation prompt with specific values. It directs the LLM to generate a list of 30 particles in a comma-separated format:

```

1 Generate a grammar list of up to 30 particles.
2
3 Your response should be a list of comma separated values, eg: 'up, down, in'
```

### A.4.2.4 Grammar Pair Generation Prompt Filled

This filled example showcases the grammar pair generation prompt, specifying a focus on particles and their grammatical usage. It instructs the LLM to generate 20 replacement error pairs:

```

1 Generate a list of up to 20 'particle' pairs highlighting grammar errors.
Focus on grammatical particles, adverb particles, discourse particles, and
negative particles.
2
3 Example/s: (look) in -> (look) at
4
5 Your response should be a list of comma separated 'particle' replacement error
pairs, eg:
6 'wrong_1 -> correct_1, wrong_2 -> correct_2'
```

## A.4.3 Prompt Manager

This section outlines the prompts used in the Prompt Manager, which coordinates text pair generation by applying error creation rules.

### A.4.3.1 Text Generation Prompt

The following prompt is a template for generating a pair of texts: one grammatically correct and one ungrammatical. It uses placeholders defined in Jinja2 syntax to specify the subject, error rules, and other details dynamically. The ungrammatical version introduces errors according to specified rules, while the correct version remains error-free:

```

1 Generate a single pair of texts about {{subject}} ({{subject_type}}): one
2 grammatically correct and one ungrammatical.
3 The ungrammatical text should be derived by applying **all** *Error Creation
4 Rules* to the correct text.
5 Each rule should (cumulatively) introduce exactly one specific grammatical
6 error, leaving the rest of the text unchanged.
7
8 The texts should be simple (short, independent sentences; no complex clauses)
9 suitable for CEFR A1-A2 learners.
10 Both texts should contain approximately {{target_word_count}} words.
11
12 *Error Creation Rules:*
13
14 {%- if errors|length == 0 and not use_code_switching %}
15     - No grammar errors provided. Return the same text as the correct and
16     incorrect one.
17 {%- endif %}
18
19 {%- for error in errors %}
20     {%- if error.type_label == "R" or error.tier in ["other", "morphology"] %}
21         - Replace {{error.category_label}} "{{error.grammar_correct}}" in the
22         correct text with "{{error.grammar_incorrect}}" to make it ungrammatical.
23         {%- elif error.type_label == "M" %}
24             - Remove {{error.category_label}} "{{error.grammar_correct}}" from the
25             correct text to make it ungrammatical (incomplete or unclear).
26         {%- elif error.type_label == "U" %}
27             - Introduce unnecessary (redundant or misplaced) {{error.category_label}}
28             {{error.category_label}} "{{error.grammar_incorrect}}" into the correct text
29             to make it ungrammatical.
30         {%- endif -%}
31 {%- endfor %}
32
33 {%- if use_code_switching %}
34     - Replace one or more words from the correct text with their German
35     equivalents (code switching) suffixed by '@g' (e.g., 'Haus@g'). Do not apply '@g'
36     to English words.
37 {%- endif %}
38
39 The output should be formatted as a JSON instance that conforms to the JSON
40 schema below:
41 """
42 {
43     "correct_text": "...",
44     "incorrect_text": "..."
45 }
46 """

```

#### A.4.3.2 Text Generation Prompt Filled: Single Error

This filled prompt demonstrates generating a pair of texts about a specific subject ("Elias"). A single grammatical error is introduced by removing the verb "listen" in the ungrammatical version:

```

1 Generate a single pair of texts about Elias (name): one grammatically correct
2 and one ungrammatical. The ungrammatical text should be derived by applying **all** *Error Creation Rules* to the correct text. Each rule should (

```

```

1 cumulatively) introduce exactly one specific grammatical error, leaving the
2 rest of the text unchanged.
3 The texts should be simple (short, independent sentences; no complex clauses)
4 suitable for CEFR A1-A2 learners. Both texts should contain approximately 12
5 words.
6
7 *Error Creation Rules:*
8     - Remove verb "listen" from the correct text to make it ungrammatical (
9     incomplete or unclear).
10
11 The output should be formatted as a JSON instance that conforms to the JSON
12 schema below:
13
14 """

```

#### A.4.3.3 Text Generation Prompt Filled: No Errors

This filled example shows a prompt where no error rules are provided. Both the correct and ungrammatical texts are identical, as no grammatical errors are introduced:

```

1 Generate a single pair of texts about Nestle (brand): one grammatically
2 correct and one ungrammatical. The ungrammatical text should be derived by
3 applying **all** *Error Creation Rules* to the correct text. Each rule should
4 (cumulatively) introduce exactly one specific grammatical error, leaving the
5 rest of the text unchanged.
6
7 The texts should be simple (short, independent sentences; no complex clauses)
8 suitable for CEFR A1-A2 learners. Both texts should contain approximately 12
9 words.
10
11 *Error Creation Rules:*
12     - No grammar errors provided. Return the same text as the correct and
13     incorrect one.
14
15 The output should be formatted as a JSON instance that conforms to the JSON
16 schema below:
17
18 """
19 {
20     "correct_text": "...",
21     "incorrect_text": "..."
22 }
23 """

```

#### A.4.3.4 Text Generation Prompt Filled: Multiple Errors and Code Switching

This filled prompt demonstrates a complex scenario where multiple errors and code switching are introduced. Errors include removing determiners, altering word order, and replacing words with their German equivalents:

```
1 Generate a single pair of texts about week (common_noun): one grammatically
2 correct and one ungrammatical. The ungrammatical text should be derived by
3 applying **all** *Error Creation Rules* to the correct text. Each rule should
4 (cumulatively) introduce exactly one specific grammatical error, leaving the
5 rest of the text unchanged.
6
7 The texts should be simple (short, independent sentences; no complex clauses)
8 suitable for CEFR A1-A2 learners. Both texts should contain approximately 16
9 words.
10
11 *Error Creation Rules:*
12   - Remove determiner "its" from the correct text to make it ungrammatical (incomplete or unclear).
13   - Replace word order "VERB NOUN ADJ" in the correct text with "NOUN VERB ADJ" to make it ungrammatical.
14   - Replace verb tense "eaten" in the correct text with "eat" to make it ungrammatical.
15   - Replace one or more words from the correct text with their German equivalents (code switching) suffixed by '@g' (e.g., 'Haus@g'). Do not apply '@g' to English words.
16
17 The output should be formatted as a JSON instance that conforms to the JSON schema below:
18
19 {
20   "correct_text": "...",
21   "incorrect_text": "..."
22 }
23
```

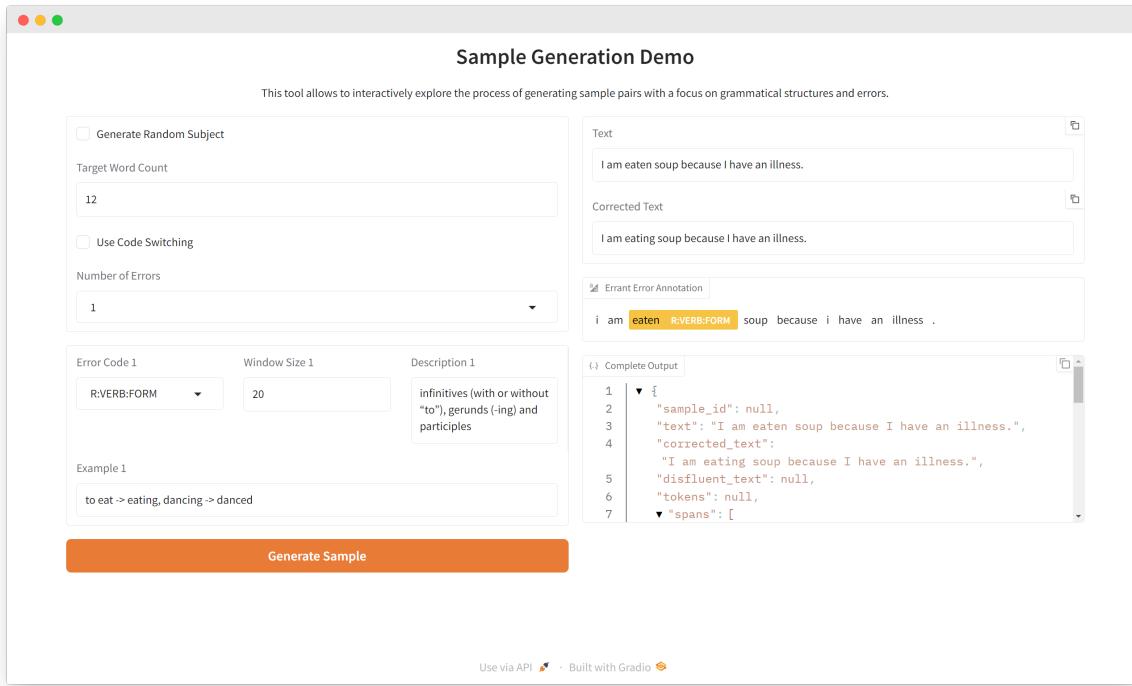
## A.5 Sample Generation Sequence Inputs

**Table A.2:** Lists the selected ERRANT error types used as input. Includes descriptions and examples used for sample generation.

Code	Description	Example
R:ADJ	Adjectives that are often confused or sound similar due to phonetic patterns	big → wide
R:ADV	Comparative or superlative adverb errors	speedily → quickly
U:CONJ		and, but, or
R:CONJ	Change in conjunction usage	and → but
M/U:DET		a, an, the
R:DET	Definite/indefinite articles, demonstratives, possessives, quantifiers, numbers, interrogatives, distributives	the → a
M/U:NOUN		cat, dog, car
R:NOUN	Nouns that share similar phonetic patterns	round → ground
M/U:PART		up, down, in
R:PART	Grammatical particles, adverb particles, discourse particles, and negative particles	(look) in → (look) at
M/U:PREP		in, on, at
R:PREP	Incorrect substitution of prepositions	of → at
M/U:PRON		she, me, them
R:PRON	Incorrect substitution of pronouns	ours → ourselves
M/:VERB		run, jump, eat
R:VERB	Misuse, substitution, or confusion of verbs, including phonetically similar forms or structurally incorrect uses	gift → give
R:MORPH	Tokens have the same lemma but nothing else in common; focus on adj → adv or adv → adj pairs	quick (adj) → quickly (adv)
R:SPELL	Misspelled words whose correction does not affect word form, tense or sentence structure	genetc → genetic, color → colour
R:WO	Incorrect word order causing ungrammatical sentence structure	ADJ NOUN → NOUN ADJ
R:ADJ:FORM	Adjective form errors (comparative/superlative)	goodest → best, bigger → biggest, more easy → easier
R:NOUN:INFL	Misuse of countable or uncountable nouns	informations → information
R:NOUN:NUM	Incorrect singular/plural form of countable nouns	cat → cats
M:VERB:FORM	Missing 'to'	(he) wants [] speak → (he) wants to speak
U:VERB:FORM	Redundant verb modification, often with auxiliary or infinitive markers like 'to'	to to eat → to eat
R:VERB:FORM	Infinitives (with or without "to"), gerunds (-ing) and participles	to eat → eating, dancing → danced
R:VERB:INFL	Misapplication of tense morphology	getted → got, fliped → flipped
R:VERB:SVA	Subject-verb agreement errors	(he) have → has, (they) barks → bark
M:VERB:TENSE	Missing auxiliary verb	(she) [] eaten → has eaten
U:VERB:TENSE	Unnecessary auxiliary verb	(she) had went to → went to
R:VERB:TENSE	Inflectional and periphrastic tense, modal verbs and passivization	eats → ate, eats → has eaten, eats → can eat, eats → was eaten

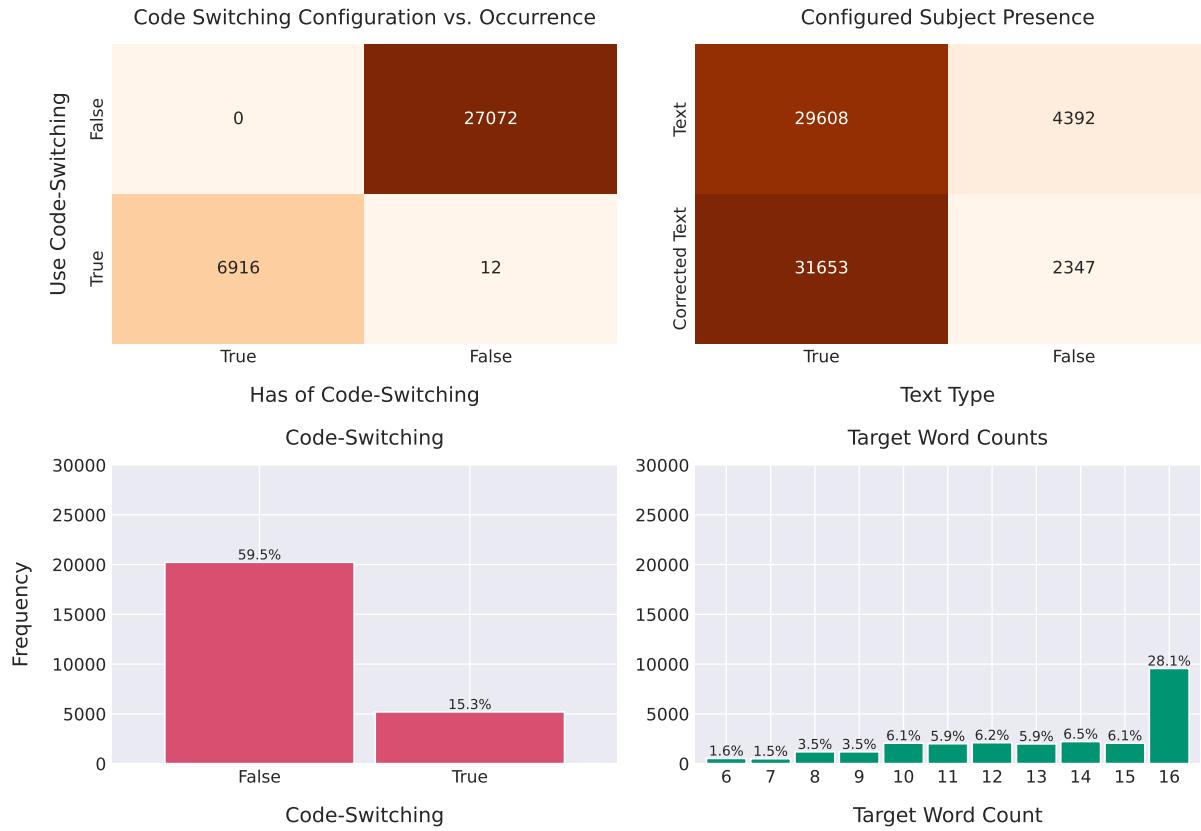
## A.6 Sample Generation App

This Gradio app as visualized in A.2 provides an interactive interface for generating sample text pairs. It allows users to customize inputs such as subject type, grammatical error details, and word count, and then visualizes the outputs with error annotations and JSON results.

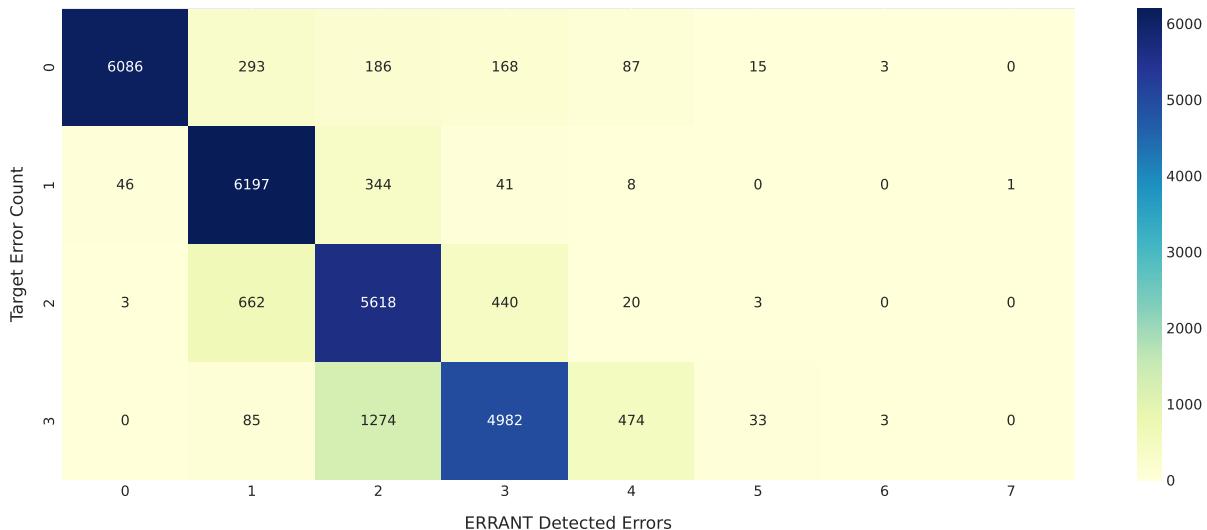


**Figure A.2:** The Sample Generation App, built using Gradio, provides an interactive platform to explore and experiment with the sample pair generation framework.

## A.7 Generated Texts Dataset Evaluation



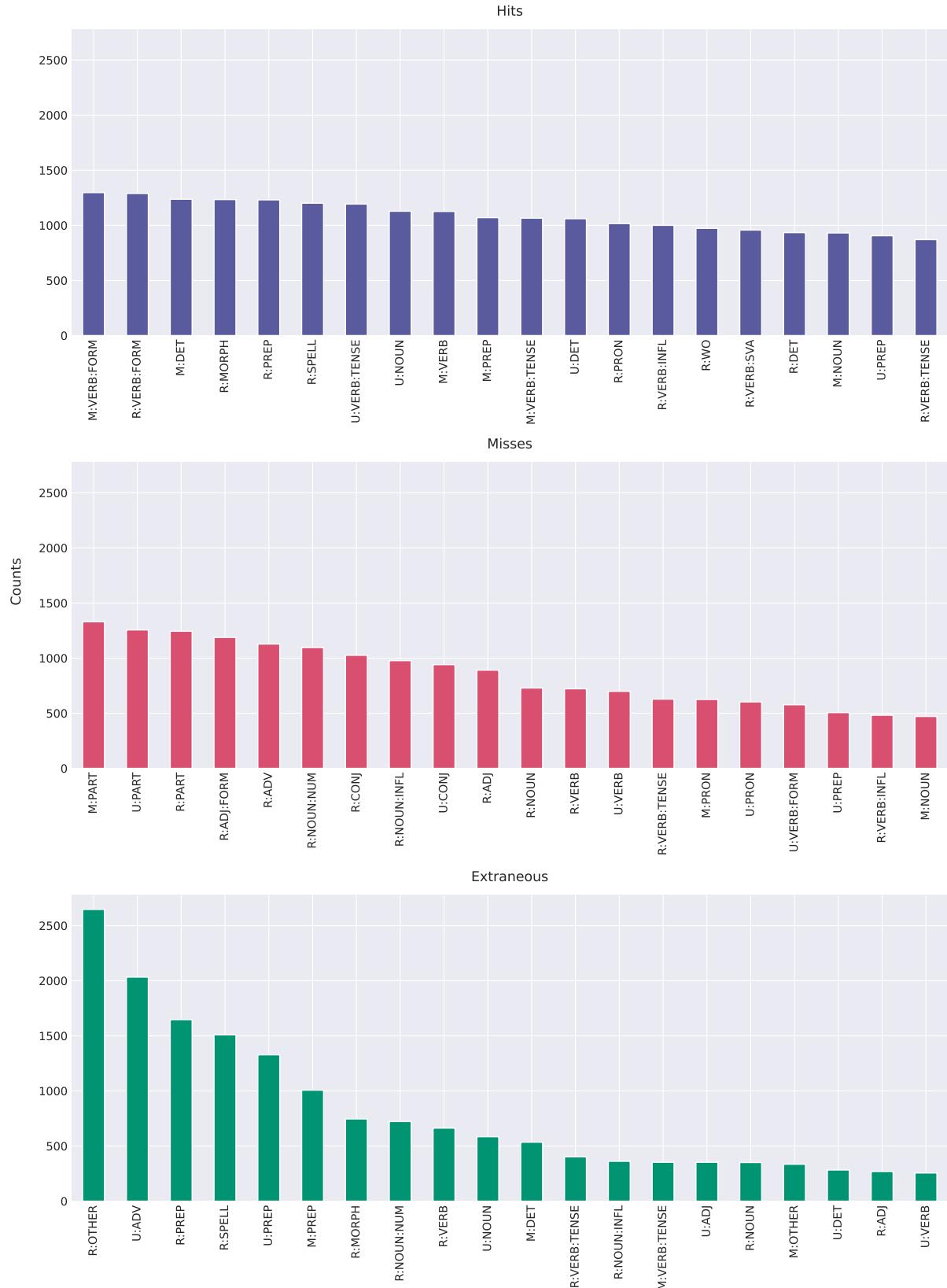
**Figure A.3:** The top row displays two heatmaps: the Code Switching Matrix (left), showing the relationship between code-switching configuration and occurrence, and Subjects in Text (right), showing whether the configured subject is in the text and corrected text. The bottom row presents two bar plots: Code-Switching Frequency (left), showing the proportion of texts configured to use code-switching; and Target Word Count Frequency (right), illustrating the distribution of configured target word counts.



**Figure A.4:** Heatmap showing the relationship between the configured number of generated errors (y-axis) and the ERRANT-detected errors (x-axis) for samples without code-switching. The chart highlights that, in most cases, the configured and detected error counts align.

**Table A.3:** Overview of configured and ERRANT-detected error counts for all error codes, including their totals and differences. The table also lists error codes excluded during generation but annotated by ERRANT. Since error types are randomly selected, they are evenly distributed. In total, 51098 errors were configured, and 47869 errors were detected.

Error Code	Error Count	Span Count	Difference
M:ADJ	0	105	-105
R:PUNCT	0	4	-4
U:ADJ	0	353	-353
R:ORTH	0	1	-1
R:WO	1396	1036	360
M:OTHER	0	335	-335
R:DET	1392	1134	258
U:PREP	1412	2234	-822
U:ADV	0	2033	-2033
R:SPELL	1405	2710	-1305
R:OTHER	0	2647	-2647
U:NOUN	1445	1713	-268
R:ADV	1378	486	892
M:VERB:FORM	1437	1347	90
R:VERB:INFL	1482	1015	467
R:PART	1425	233	1192
R:NOUN:NUM	1415	1042	373
U:PRON	1377	809	568
U:VERB:FORM	1415	870	545
R:PREP	1422	2878	-1456
R:MORPH	1477	1980	-503
U:CONJ	1404	469	935
R:ADJ:FORM	1431	270	1161
U:VERB	1419	976	443
R:VERB	1438	1379	59
U:DET	1420	1343	77
U:VERB:TENSE	1441	1419	22
R:VERB:SVA	1372	998	374
R:PRON	1343	1052	291
R:VERB:FORM	1474	1489	-15
M:CONTR	0	9	-9
R:VERB:TENSE	1499	1273	226
U:PUNCT	0	4	-4
M:PRON	1437	828	609
U:PART	1411	362	1049
R:CONJ	1397	381	1016
U:OTHER	0	178	-178
R:ADJ	1403	781	622
R:NOUN:INFL	1418	803	615
M:PUNCT	0	7	-7
M:CONJ	0	16	-16
M:DET	1425	1772	-347
M:NOUN	1402	1039	363
M:PREP	1389	2077	-688
M:PART	1464	192	1272
M:ADV	0	104	-104
M:VERB:TENSE	1375	1418	-43
M:VERB	1401	1186	215
R:NOUN	1457	1079	378



**Figure A.5:** Bar plots illustrating the distribution of configured and detected errors in the generated dataset: Hits (errors configured and detected), Misses (errors configured but not detected), Extraneous (errors not configured but detected), and Substitutions (errors configured but detected as). The y-axis represents counts for each category.

	M:DET	M:PREP	M:VERB:TENSE	R:MORPH	R:NOUN:INFL	R:NOUN:NUM	R:OTHER	R:PREP	R:SPEL	R:VERB	R:VERB:TENSE	U:ADV	U:DET	U:NOUN	U:PREP
Detected															
M:PART	0	974	0	0	0	0	0	0	0	0	0	0	0	0	0
M:PRON	383	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M:VERB:TENSE	0	0	0	0	0	0	226	0	0	0	0	0	0	0	0
R:ADJ	0	0	0	572	0	0	0	0	0	0	0	0	0	0	0
R:ADJ:FORM	0	0	0	0	0	0	0	0	0	0	0	890	0	0	0
R:ADV	0	0	0	0	0	0	307	0	494	0	0	401	0	0	0
R:CONJ	0	0	0	0	0	0	379	573	0	0	0	0	0	0	0
R:NOUN	0	0	0	0	0	0	0	0	293	0	0	0	0	0	0
R:NOUN:INFL	0	0	0	0	0	672	0	0	0	0	0	0	0	0	0
R:NOUN:NUM	0	0	0	0	356	0	0	0	268	0	0	0	0	0	0
R:PART	0	0	0	0	0	0	0	1031	0	0	0	0	0	0	0
R:VERB	0	0	0	0	0	0	0	0	0	0	364	0	0	0	0
R:VERB:INFL	0	0	0	0	0	0	0	0	0	246	0	0	0	0	0
R:VERB:SVA	0	0	0	0	0	0	310	0	0	0	0	0	0	0	0
R:VERB:TENSE	0	0	347	0	0	0	0	0	0	0	0	0	0	0	0
U:CONJ	0	0	0	0	0	0	0	0	0	0	0	253	0	0	479
U:PART	0	0	0	0	0	0	0	0	0	0	0	236	0	0	791
U:PRON	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0
U:VERB	0	0	0	0	0	0	0	0	0	0	0	0	0	385	0

**Figure A.6:** Confusion matrix illustrating the relationships between configured and ERRANT-detected error pairs in the dataset. It highlights common error type confusions occurring during the generation and annotation process. Rows represent the configured error types, and columns represent the detected error types. Only pairs with more than 200 occurrences are shown, and rows/columns with values have been removed for clarity.

# B Audio Generation

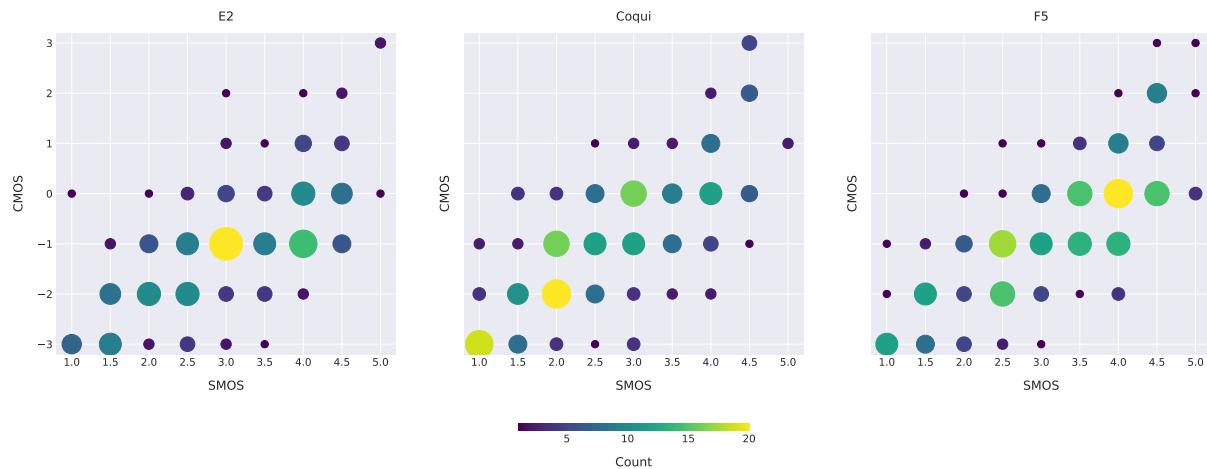
## B.1 Evaluation App

The Evaluation App interface consists of two main sections:

- Landing Page (Top Screenshot):** This section contains instructions for evaluating TTS systems. It explains that users will compare audio samples generated by different TTS systems against ground truth speech. It details the SMOS (Similarity Mean Opinion Score) and CMOS (Comparative Mean Opinion Score) scales.
- Evaluation Process (Bottom Screenshot):** This section shows a step-by-step evaluation process. It includes:
  - A timeline showing "Reference Audio" and "Generated Audio" side-by-side for comparison. The reference audio duration is 0:10 and the generated audio duration is 0:14.
  - A text input field labeled "Synthesized Text" containing the sentence "I like the beach. I like the beach, yes. Because".
  - A "SMOS (Similarity Mean Opinion Score) (1 to 5)" scale with options from 1.0 to 5.0.
  - A "CMOS (Comparative Mean Opinion Score) (-3 to 3)" scale with options from -3 to 3.
  - A "Submit" button at the bottom.

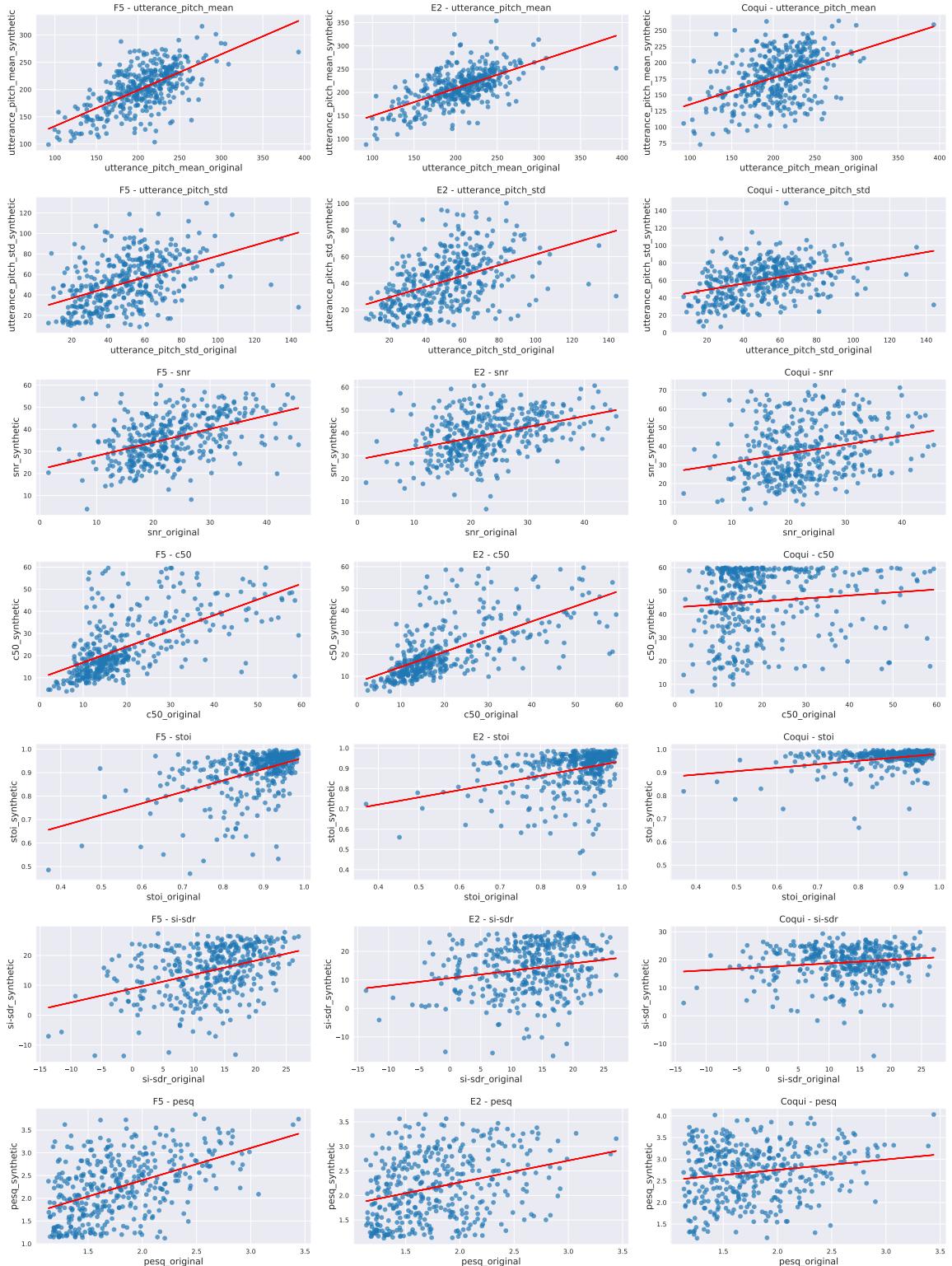
**Figure B.1:** The Evaluation App, built using Gradio, provides an interface for audio sample evaluation. The top screenshot shows the landing page with instructions, while the bottom screenshot demonstrates the step-by-step evaluation process. At each step, a real and a synthetic audio sample are presented for listening and scoring using SMOS and CMOS.

## B.2 TTS System Evaluation



**Figure B.2:** Comparison of SMOS (Subjective Mean Opinion Score) and CMOS (Comparison Mean Opinion Score) for three TTS systems. The size of the points reflects the number of occurrences, and the color indicates the count density.

### B.3 Dataspeech Evaluation



**Figure B.3:** Scatter plots with fitted regression lines illustrating the alignment between DataSpeech annotations of reference audio and synthetic audio feature values for each dataset. Each subplot represents a different numerical feature across multiple datasets, with data points (blue) and the corresponding linear regression fit (red). Higher alignment between original and synthetic values indicates better preservation of speech characteristics by the TTS system.

# C Training Appendix

## C.1 Performance of Human vs. Automatic ERRANT Error Detection

**Table C.1:** This table compares human-annotated errors with automatically generated annotations. Only data from the annotated subset (Section 4.1.2) is included. The automatically generated annotations were produced using GEC and ERRANT, based on comparisons between the original texts and their corrected versions. The evaluation metrics highlight the differences between human and automated annotation. The low scores highlight the challenges and limitations of the proposed methods for error detection.

Strategy	Precision	Recall	F1 Score
Simple	0.325	0.175	0.227
Multi_Union	0.143	0.305	0.195
Multi_Error_Majority	0.366	0.166	0.228
Multi_Span_Majority	0.339	0.170	0.227
Multi_Prob	0.321	0.193	0.241

## C.2 Error Type Evaluation on Synth Data

**Table C.2:** This table compares detected errors in ASR model predictions (on synth test data) using GEC and ERRANT with: a) the predefined error types used as inputs for text synthesis (“Errors”), and b) the ERRANT-detected errors between the correct and incorrect generated text versions (“Spans”). The low scores highlight the challenges and limitations of the proposed methods for error detection. Some models can retain up to 30% of the errors in a synthesized sample, meaning that these errors persist through both the TTS and STT processes.

Evaluation	Experiment	Errors			Spans		
		F1 Score	Precision	Recall	F1 Score	Precision	Recall
synth_0_20	simple	0.177	0.2	0.158	0.24	0.268	0.217
synth_0_20	multi_union	0.206	0.188	0.227	0.277	0.25	0.31
synth_0_20	multi_error_majority	0.178	0.209	0.156	0.245	0.283	0.216
synth_0_20	multi_span_majority	0.178	0.206	0.158	0.245	0.278	0.218
synth_0_20	multi_prob	0.184	0.201	0.169	0.252	0.272	0.234
synth_0_40	simple	0.2	0.223	0.181	0.271	0.301	0.247
synth_0_40	multi_union	0.225	0.207	0.246	0.308	0.282	0.34
synth_0_40	multi_error_majority	0.198	0.231	0.173	0.268	0.309	0.236
synth_0_40	multi_span_majority	0.199	0.229	0.176	0.267	0.305	0.238
synth_0_40	multi_prob	0.203	0.22	0.188	0.279	0.3	0.26
synth_0_60	simple	0.196	0.225	0.173	0.293	0.331	0.262
synth_0_60	multi_union	0.222	0.209	0.236	0.322	0.3	0.348
synth_0_60	multi_error_majority	0.201	0.248	0.17	0.292	0.353	0.249
synth_0_60	multi_span_majority	0.199	0.238	0.17	0.288	0.339	0.25
synth_0_60	multi_prob	0.201	0.226	0.181	0.301	0.333	0.275
synth_0_80	simple	0.213	0.25	0.185	0.3	0.35	0.262
synth_0_80	multi_union	0.242	0.23	0.255	0.325	0.307	0.345
synth_0_80	multi_error_majority	0.216	0.267	0.181	0.298	0.365	0.251
synth_0_80	multi_span_majority	0.215	0.259	0.184	0.295	0.353	0.254
synth_0_80	multi_prob	0.22	0.251	0.196	0.306	0.346	0.274
synth_60_0	simple	0.196	0.225	0.173	0.293	0.331	0.262
synth_60_0	multi_union	0.222	0.209	0.236	0.322	0.3	0.348
synth_60_0	multi_error_majority	0.201	0.248	0.17	0.292	0.353	0.249
synth_60_0	multi_span_majority	0.199	0.238	0.17	0.288	0.339	0.25
synth_60_0	multi_prob	0.201	0.226	0.181	0.301	0.333	0.275
synth_60_20	simple	0.194	0.237	0.165	0.276	0.333	0.236
synth_60_20	multi_union	0.227	0.216	0.24	0.324	0.305	0.346
synth_60_20	multi_error_majority	0.199	0.251	0.165	0.284	0.354	0.238
synth_60_20	multi_span_majority	0.201	0.248	0.17	0.287	0.349	0.244
synth_60_20	multi_prob	0.204	0.239	0.177	0.291	0.337	0.256
synth_60_40	simple	0.199	0.24	0.169	0.279	0.331	0.241
synth_60_40	multi_union	0.221	0.211	0.233	0.311	0.293	0.333
synth_60_40	multi_error_majority	0.194	0.243	0.161	0.281	0.346	0.237
synth_60_40	multi_span_majority	0.195	0.238	0.164	0.281	0.338	0.24
synth_60_40	multi_prob	0.203	0.237	0.177	0.291	0.334	0.257
synth_60_60	simple	0.212	0.256	0.182	0.293	0.348	0.253
synth_60_60	multi_union	0.234	0.224	0.244	0.319	0.303	0.337
synth_60_60	multi_error_majority	0.211	0.267	0.175	0.293	0.365	0.245
synth_60_60	multi_span_majority	0.21	0.258	0.177	0.293	0.355	0.249
synth_60_60	multi_prob	0.211	0.247	0.184	0.296	0.342	0.261
synth_60_80	simple	0.202	0.239	0.175	0.291	0.342	0.254
synth_60_80	multi_union	0.231	0.219	0.245	0.322	0.303	0.344
synth_60_80	multi_error_majority	0.201	0.25	0.169	0.283	0.348	0.238
synth_60_80	multi_span_majority	0.208	0.252	0.177	0.289	0.347	0.247
synth_60_80	multi_prob	0.204	0.235	0.18	0.295	0.337	0.262

### C.3 Error Type Evaluation on Real Data

**Table C.3:** This table compares detected errors in ASR model predictions using GEC and ERRANT with the human-annotated subset of the ChaLL data on real test data. The low scores highlight the challenges of this approach on real data, regardless of the detection strategy, making it impractical for detailed error evaluation.

Evaluation	Experiment	Errors		
		F1 Score	Precision	Recall
0_20	simple	0.036	0.027	0.054
0_20	multi_union	0.037	0.024	0.086
0_20	multi_error_majority	0.034	0.026	0.048
0_20	multi_span_majority	0.033	0.025	0.048
0_20	multi_prob	0.037	0.027	0.059
0_40	simple	0.03	0.022	0.049
0_40	multi_union	0.036	0.023	0.086
0_40	multi_error_majority	0.039	0.029	0.059
0_40	multi_span_majority	0.038	0.028	0.059
0_40	multi_prob	0.038	0.027	0.065
0_60	simple	0.051	0.038	0.076
0_60	multi_union	0.036	0.023	0.081
0_60	multi_error_majority	0.042	0.032	0.059
0_60	multi_span_majority	0.041	0.031	0.059
0_60	multi_prob	0.047	0.034	0.076
0_80	simple	0.032	0.024	0.049
0_80	multi_union	0.034	0.022	0.076
0_80	multi_error_majority	0.039	0.031	0.054
0_80	multi_span_majority	0.038	0.029	0.054
0_80	multi_prob	0.034	0.025	0.054
60_0	simple	0.042	0.036	0.053
60_0	multi_union	0.06	0.042	0.105
60_0	multi_error_majority	0.038	0.033	0.046
60_0	multi_span_majority	0.042	0.035	0.053
60_0	multi_prob	0.04	0.033	0.053
60_20	simple	0.057	0.051	0.066
60_20	multi_union	0.062	0.044	0.105
60_20	multi_error_majority	0.059	0.054	0.066
60_20	multi_span_majority	0.058	0.052	0.066
60_20	multi_prob	0.055	0.047	0.066
60_40	simple	0.054	0.045	0.067
60_40	multi_union	0.052	0.036	0.093
60_40	multi_error_majority	0.05	0.043	0.06
60_40	multi_span_majority	0.05	0.043	0.06
60_40	multi_prob	0.052	0.042	0.067
60_60	simple	0.047	0.04	0.056
60_60	multi_union	0.052	0.037	0.088
60_60	multi_error_majority	0.043	0.038	0.05
60_60	multi_span_majority	0.043	0.038	0.05
60_60	multi_prob	0.045	0.038	0.056
60_80	simple	0.049	0.042	0.058
60_80	multi_union	0.062	0.045	0.104
60_80	multi_error_majority	0.047	0.042	0.052
60_80	multi_span_majority	0.046	0.041	0.052
60_80	multi_prob	0.047	0.04	0.058

# Declarations

This chapter outlines the AI tools used in this study and their respective purposes. ChatGPT (<https://chatgpt.com/>) from OpenAI was used for text improvement tasks such as paraphrasing self-authored English content and correcting grammatical issues. Additionally, ChatGPT was employed to help with code documentation, further enhancing the clarity and comprehensibility of the codebase. Elicit (<https://elicit.org/>) was used to assist in extracting key information from academic papers and organizing relevant literature.

# Bibliography

- Achanta, S., Godambe, T., & Gangashetty, S. V. (2015). An investigation of recurrent neural network architectures for statistical parametric speech synthesis, 859–863. <https://doi.org/10.21437/Interspeech.2015-266>
- Ahmed, B., Ballard, K., Burnham, D., Sirojan, T., Mehmood, H., Estival, D., Baker, E., Cox, F., Arciuli, J., Binders, T., et al. (2021). AusKidTalk: An auditory-visual corpus of 3-to 12-year-old Australian children's speech. *Annual Conference of the International Speech Communication Association* (22nd: 2021), 3680–3684.
- Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F. M., & Weber, G. (2020, March 5). *Common Voice: A Massively-Multilingual Speech Corpus*. arXiv: 1912.06670 [cs]. <https://doi.org/10.48550/arXiv.1912.06670>
- Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., von Platen, P., Saraf, Y., Pino, J., Baevski, A., Conneau, A., & Auli, M. (2021, December 16). XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale. arXiv: 2111.09296 [cs, eess]. <https://doi.org/10.48550/arXiv.2111.09296>
- Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *Advances in Neural Information Processing Systems*, 33, 12449–12460. Retrieved August 8, 2023, from <https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html>
- Batliner, A., Blomberg, M., D'Arcy, S., Elenius, D., Giuliani, D., Gerosa, M., Hacker, C., Russell, M., Steidl, S., & Wong, M. (2005). The PF\_STAR children's speech corpus.
- Baur, C., Caines, A., Chua, C., Gerlach, J., Qian, M., Rayner, M., Russell, M., Strik, H., & Wei, X. (2019). Overview of the 2019 spoken call shared task.
- Betker, J. (2023). *Better speech synthesis through scaling*.
- Boigne, J. (2021, September 30). *An Illustrated Tour of Wav2vec 2.0*. Jonathan Bgn. Retrieved December 12, 2023, from <https://jonathanbgn.com/2021/09/30/illustrated-wav2vec-2.html>
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., . . . Amodei, D. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901. Retrieved December 30, 2024, from <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html>
- Bryant, C., Felice, M., & Briscoe, T. (2017, July). Automatic Annotation and Evaluation of Error Types for Grammatical Error Correction. In R. Barzilay & M.-Y. Kan (Eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 793–805). Association for Computational Linguistics. <https://doi.org/10.18653/v1/P17-1074>
- Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., & Zhang, Y. (2023, April 13). *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. arXiv: 2303.12712 [cs]. <https://doi.org/10.48550/arXiv.2303.12712>
- Burns, A. (2019). Concepts for Teaching Speaking in the English Language Classroom. *LEARN Journal: Language Education and Acquisition Research Network*, 12(1), 1–11. Retrieved December 26, 2024, from <https://eric.ed.gov/?id=EJ1225673>  
ERIC Number: EJ1225673.

- Casanova, E., Davis, K., Gölge, E., Göknar, G., Gulea, I., Hart, L., Aljafari, A., Meyer, J., Morais, R., Olayemi, S., & Weber, J. (2024, June 7). *XTTS: A Massively Multilingual Zero-Shot Text-to-Speech Model*. arXiv: 2406.04904 [eess]. <https://doi.org/10.48550/arXiv.2406.04904>
- Casanova, E., Shulby, C., Gölge, E., Müller, N. M., De Oliveira, F. S., Junior, A. C., Soares, A. d. S., Aluisio, S. M., & Ponti, M. A. (2021). *SC-GlowTTS: An efficient zero-shot multi-speaker text-to-speech model*.
- Chen, Y., Niu, Z., Ma, Z., Deng, K., Wang, C., Zhao, J., Yu, K., & Chen, X. (2024, October 15). *F5-TTS: A Fairytaler that Fakes Fluent and Faithful Speech with Flow Matching*. arXiv: 2410.06885 [eess]. <https://doi.org/10.48550/arXiv.2410.06885>
- Cornell, S., Daresky, J., Duan, Z., & Watanabe, S. (2024, August 17). *Generating Data with Text-to-Speech and Large-Language Models for Conversational Speech Recognition*. arXiv: 2408.09215 [eess]. <https://doi.org/10.48550/arXiv.2408.09215>
- Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2011). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1), 30–42.
- Das, A., Li, J., Zhao, R., & Gong, Y. (2018, March 15). *Advancing Connectionist Temporal Classification With Attention Modeling*. arXiv.org. Retrieved October 10, 2023, from <https://arxiv.org/abs/1803.05563v1>
- Deriu, J., Rodrigo, A., Otegi, A., Echegoyen, G., Rosset, S., Agirre, E., & Cieliebak, M. (2021). Survey on Evaluation Methods for Dialogue Systems. *Artificial Intelligence Review*, 54(1), 755–810. <https://doi.org/10.1007/s10462-020-09866-x>
- Ellis, R. (2021). Explicit and implicit oral corrective feedback. *The Cambridge handbook of corrective feedback in second language learning and teaching*, 341–364.
- Eskenazi, M. S. (1996). *Kids: A database of children's speech* [Doctoral dissertation, Acoustical Society of America].
- Eskimez, S. E., Wang, X., Thakker, M., Li, C., Tsai, C.-H., Xiao, Z., Yang, H., Zhu, Z., Tang, M., Tan, X., Liu, Y., Zhao, S., & Kanda, N. (2024, September 12). *E2 TTS: Embarrassingly Easy Fully Non-Autoregressive Zero-Shot TTS*. arXiv: 2406.18009 [eess]. <https://doi.org/10.48550/arXiv.2406.18009>
- Felice, M., Bryant, C., & Briscoe, T. (2016, December). Automatic Extraction of Learner Errors in ESL Sentences Using Linguistically Enhanced Alignments. In Y. Matsumoto & R. Prasad (Eds.), *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (pp. 825–835). The COLING 2016 Organizing Committee. Retrieved January 4, 2025, from <https://aclanthology.org/C16-1079/>
- Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., & Hovy, E. (2021, December 1). *A Survey of Data Augmentation Approaches for NLP*. arXiv: 2105.03075 [cs]. <https://doi.org/10.48550/arXiv.2105.03075>
- Gao, Y., Morioka, N., Zhang, Y., & Chen, N. (2023). E3 TTS: Easy end-to-end diffusion-based text to speech. *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 1–8. <https://doi.org/10.1109/ASRU57964.2023.10389766>
- Gerosa, M., Giuliani, D., & Narayanan, S. (2006). Acoustic analysis and automatic recognition of spontaneous children's speech. *Ninth International Conference on Spoken Language Processing*.
- Gerosa, M., Giuliani, D., Narayanan, S., & Potamianos, A. (2009). A review of ASR technologies for children's speech. *Proceedings of the 2nd Workshop on Child, Computer and Interaction*, 1–8.
- Gretter, R., Matassoni, M., Bannò, S., & Falavigna, D. (2020). *TLT-school: A corpus of non native children speech*.

- Grimm, N., Meyer, M., & Volkmann, L. (2015). *Teaching english*. Narr Francke Attempto Verlag.
- Gurunath Shivakumar, P., & Georgiou, P. (2020). Transfer learning from adult to children for speech recognition: Evaluation, analysis and recommendations. *Computer Speech & Language*, 63, 101077. <https://doi.org/10.1016/j.csl.2020.101077>
- Hagen, A., Pellom, B., & Cole, R. (2003). Children's speech recognition with application to interactive books and tutors. *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No. 03EX721)*, 186–191.
- Hedge, T. (2001). *Teaching and learning in the language classroom* (Vol. 106). Oxford university press Oxford.
- Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6), 82–97.
- Hovy, D., Berg-Kirkpatrick, T., Vaswani, A., & Hovy, E. (2013). Learning whom to trust with MACE. *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1120–1130.
- Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., & Mohamed, A. (2021, June 14). *HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units*. arXiv: 2106.07447 [cs, eess]. Retrieved September 19, 2023, from <http://arxiv.org/abs/2106.07447>
- Hürlimann, M., Sauer, L., Schneider, G., Graën, J., Goldman, J.-P., Michot, J., Mlynchyk, K., Uluslu, A. Y., Stroescu, I.-C., Deriu, J., Geiss, M., & Cieliebak, M. (2024, June). ChaLL - A Chatbot for Language Learners. In C. Corsin, C. Mark, W. Albert, M. Claudiu, M. Elisabeth, & Z. Lucas (Eds.), *Proceedings of the 9th edition of the Swiss Text Analytics Conference* (pp. 168–168). Association for Computational Linguistics. Retrieved December 27, 2024, from <https://aclanthology.org/2024.swisstext-1.20>
- Islam, R., & Moushi, O. M. (2024). GPT-4o: The Cutting-Edge Advancement in Multimodal LLM. Retrieved January 5, 2025, from <https://www.authorea.com/users/771522/articles/1121145-gpt-4o-the-cutting-edge-advancement-in-multimodal-lm>
- Jain, R., Barcovschi, A., Yiwere, M., Corcoran, P., & Cucu, H. (2023, July 24). *Adaptation of Whisper models to child speech recognition*. arXiv: 2307.13008 [cs, eess]. <https://doi.org/10.48550/arXiv.2307.13008>
- Jain, R., Yiwere, M. Y., Bigioi, D., Corcoran, P., & Cucu, H. (2022). A Text-to-Speech Pipeline, Evaluation Methodology, and Initial Fine-Tuning Results for Child Speech Synthesis. *IEEE Access*, 10, 47628–47642. <https://doi.org/10.1109/ACCESS.2022.3170836>
- Jang, E., Gu, S., & Poole, B. (2017, August 5). *Categorical Reparameterization with Gumbel-Softmax*. arXiv: 1611.01144 [cs, stat]. <https://doi.org/10.48550/arXiv.1611.01144>
- Jia, Y., Zhang, Y., Weiss, R., Wang, Q., Shen, J., Ren, F., Nguyen, P., Pang, R., Lopez Moreno, I., Wu, Y., et al. (2018). Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *Advances in neural information processing systems*, 31.
- Ju, Z., Wang, Y., Shen, K., Tan, X., Xin, D., Yang, D., Liu, Y., Leng, Y., Song, K., Tang, S., Wu, Z., Qin, T., Li, X.-Y., Ye, W., Zhang, S., Bian, J., He, L., Li, J., & Zhao, S. (2024, April 23). *NaturalSpeech 3: Zero-Shot Speech Synthesis with Factorized Codec and Diffusion Models*. arXiv: 2403.03100 [eess]. <https://doi.org/10.48550/arXiv.2403.03100>
- Kaur, N., & Singh, P. (2023). Conventional and contemporary approaches used in text to speech synthesis: A review. *Artificial Intelligence Review*, 56(7), 5837–5880. <https://doi.org/10.1007/s10462-022-10315-0>

- Khan, R. A., & Chitode, J. S. (2016). Concatenative speech synthesis: A review. *International Journal of Computer Applications*, 136(3), 1–6.
- Kim, J., Kim, S., Kong, J., & Yoon, S. (2020). Glow-TTS: A Generative Flow for Text-to-Speech via Monotonic Alignment Search. *Advances in Neural Information Processing Systems*, 33, 8067–8077. Retrieved December 31, 2024, from <https://proceedings.neurips.cc/paper/2020/hash/5c3b99e8f92532e5ad1556e53ceea00c-Abstract.html>
- Kim, S., Hori, T., & Watanabe, S. (2017). Joint CTC-attention based end-to-end speech recognition using multi-task learning. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4835–4839.
- Lacombe, Y., Srivastav, V., & Gandhi, S. (2024). Data-speech. *GitHub*. <https://github.com/ylacombe/dataspeech>
- Le, M., Vyas, A., Shi, B., Karrer, B., Sari, L., Moritz, R., Williamson, M., Manohar, V., Adi, Y., Mahadeokar, J., & Hsu, W.-N. (2023, October 19). Voicebox: Text-Guided Multilingual Universal Speech Generation at Scale. arXiv: 2306.15687 [eess]. <https://doi.org/10.48550/arXiv.2306.15687>
- Le Scao, T., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., et al. (2023). Bloom: A 176b-parameter open-access multilingual language model.
- Lee, S., Potamianos, A., & Narayanan, S. (1999a). Acoustics of children's speech: Developmental changes of temporal and spectral parameters. *The Journal of the Acoustical Society of America*, 105(3), 1455–1468.
- Lee, S., Potamianos, A., & Narayanan, S. (1999b). Acoustics of children's speech: Developmental changes of temporal and spectral parameters. *The Journal of the Acoustical Society of America*, 105(3), 1455–1468. <https://doi.org/10.1121/1.426686>
- Le-Khac, P. H., Healy, G., & Smeaton, A. F. (2020). Contrastive Representation Learning: A Framework and Review. *IEEE Access*, 8, 193907–193934. <https://doi.org/10.1109/ACCESS.2020.3031549>
- Li, J. (2021, November 2). *Recent Advances in End-to-End Automatic Speech Recognition*. arXiv.org. Retrieved October 10, 2023, from <https://arxiv.org/abs/2111.01690v2>
- Ling, Z.-H., Kang, S.-Y., Zen, H., Senior, A., Schuster, M., Qian, X.-J., Meng, H. M., & Deng, L. (2015). Deep Learning for Acoustic Modeling in Parametric Speech Generation: A systematic review of existing techniques and future trends. *IEEE Signal Processing Magazine*, 32(3), 35–52. <https://doi.org/10.1109/MSP.2014.2359987>
- Lu, R., Shahin, M., & Ahmed, B. (2022, November 14). *Improving Children's Speech Recognition by Fine-tuning Self-supervised Adult Speech Representations*. arXiv: 2211.07769 [cs, eess]. <https://doi.org/10.48550/arXiv.2211.07769>
- Lyth, D., & King, S. (2024, February 2). *Natural language guidance of high-fidelity text-to-speech with synthetic annotations*. arXiv: 2402.01912 [cs]. <https://doi.org/10.48550/arXiv.2402.01912>
- Malik, M., Malik, M. K., Mehmood, K., & Makhdoom, I. (2021). Automatic speech recognition: A survey. *Multimedia Tools and Applications*, 80(6), 9411–9457. <https://doi.org/10.1007/s11042-020-10073-7>
- Mehta, S., Tu, R., Beskow, J., Székely, É., & Henter, G. E. (2024, January 9). Matcha-TTS: A fast TTS architecture with conditional flow matching. arXiv: 2309.03199 [eess]. <https://doi.org/10.48550/arXiv.2309.03199>
- Miao, H., Cheng, G., Zhang, P., Li, T., & Yan, Y. (2019). Online Hybrid CTC/Attention Architecture for End-to-End Speech Recognition. *Interspeech*, 2623–2627.
- Michot, J. (2024). *Automatic Speech Processing for Language Learners*.

- Michot, J., Hürlimann, M., Deriu, J., Sauer, L., Mlynchyk, K., & Cieliebak, M. (2024, June 5). *Error-preserving Automatic Speech Recognition of Young English Learners' Language*. arXiv: 2406.03235 [cs]. <https://doi.org/10.48550/arXiv.2406.03235>
- Ni, J., Young, T., Pandelea, V., Xue, F., & Cambria, E. (2022, March 30). *Recent Advances in Deep Learning Based Dialogue Systems: A Systematic Survey*. arXiv: 2105.04387 [cs]. <https://doi.org/10.48550/arXiv.2105.04387>
- Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016, September 19). *WaveNet: A Generative Model for Raw Audio*. arXiv: 1609.03499 [cs]. <https://doi.org/10.48550/arXiv.1609.03499>
- OpenAI. (2022, November). Introducing ChatGPT. <https://openai.com/blog/chatgpt>
- OpenAI. (2023a). GPT-4 technical report. <https://openai.com/research/gpt-4>
- OpenAI. (2023b). Hello GPT-4o. <https://openai.com/index/hello-gpt-4o/>
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35, 27730–27744. Retrieved December 30, 2024, from [https://proceedings.neurips.cc/paper\\_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html](https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html)
- P.862.2 : Wideband extension to Recommendation P.862 for the assessment of wideband telephone networks and speech codecs. (2005). Retrieved January 29, 2025, from <https://www.itu.int/rec/T-REC-P.862.2-200711-W/en>
- Pakula, H.-M. (2019). Teaching speaking. *Apples - Journal of Applied Language Studies*, 13(1), 95–111. <https://doi.org/10.17011/apples/urn.201903011691>
- Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). Librispeech: An ASR corpus based on public domain audio books. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5206–5210. <https://doi.org/10.1109/ICASSP.2015.7178964>
- Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002, July). Bleu: A Method for Automatic Evaluation of Machine Translation. In P. Isabelle, E. Charniak, & D. Lin (Eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 311–318). Association for Computational Linguistics. <https://doi.org/10.3115/1073083.1073135>
- Park, J., Park, C., & Lim, H. (2024, June 11). *ChatLang-8: An LLM-Based Synthetic Data Generation Framework for Grammatical Error Correction*. arXiv: 2406.03202 [cs]. <https://doi.org/10.48550/arXiv.2406.03202>
- Pfenninger, S. E., & Lendl, J. (2017). Transitional woes: On the impact of L2 input continuity from primary to secondary school. *Studies in Second Language Learning and Teaching*, 7(3), 443–469. <https://doi.org/10.14746/ssllt.2017.7.3.5>
- Ping, W., Peng, K., Gibiansky, A., Arik, S. O., Kannan, A., Narang, S., Raiman, J., & Miller, J. (2018, February 22). Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning. arXiv: 1710.07654 [cs]. <https://doi.org/10.48550/arXiv.1710.07654>
- Pinker, S. (2003). *The language instinct: How the mind creates language*. Penguin uK.
- Popović, M. (2015). chrF: Character n-gram F-score for automatic MT evaluation. *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 392–395. <https://doi.org/10.18653/v1/W15-3049>
- Potamianos, A., & Narayanan, S. (2003). Robust recognition of children's speech. *IEEE Transactions on Speech and Audio Processing*, 11(6), 603–616. <https://doi.org/10.1109/TSA.2003.818026>

- Potter, T., & Yuan, Z. (2024, October 14). *LLM-based Code-Switched Text Generation for Grammatical Error Correction*. arXiv: 2410.10349 [cs]. <https://doi.org/10.48550/arXiv.2410.10349>
- Prabhavalkar, R., Hori, T., Sainath, T. N., Schlüter, R., & Watanabe, S. (2023, March 2). *End-to-End Speech Recognition: A Survey*. arXiv: 2303.03329 [cs, eess]. <https://doi.org/10.48550/arXiv.2303.03329>
- Pradhan, S., Cole, R., & Ward, W. (2016). My science tutor—learning science with a conversational virtual tutor. *Proceedings of ACL-2016 System Demonstrations*, 121–126.
- Qian, Y., Fan, Y., Hu, W., & Soong, F. K. (2014). On the training aspects of Deep Neural Network (DNN) for parametric TTS synthesis. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3829–3833. <https://doi.org/10.1109/ICASSP.2014.6854318>
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 9.
- Radford, A. (2018). Improving language understanding by generative pre-training.
- Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022, December 6). *Robust Speech Recognition via Large-Scale Weak Supervision*. arXiv: 2212.04356 [cs, eess]. <https://doi.org/10.48550/arXiv.2212.04356>
- Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.
- Rao, P. S. (2019). The importance of speaking skills in English classrooms. *Alford Council of International English & Literature Journal (ACIELJ)*, 2(2), 6–18.
- Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., & Liu, T.-Y. (2022, August 8). *FastSpeech 2: Fast and High-Quality End-to-End Text to Speech*. arXiv: 2006.04558 [eess]. <https://doi.org/10.48550/arXiv.2006.04558>
- Ren, Y., Ruan, Y., Tan, X., Qin, T., Zhao, S., Zhao, Z., & Liu, T.-Y. (2019). FastSpeech: Fast, Robust and Controllable Text to Speech. *Advances in Neural Information Processing Systems*, 32. Retrieved December 31, 2024, from [https://proceedings.neurips.cc/paper\\_files/paper/2019/hash/f63f65b503e22cb970527f23c9ad7db1-Abstract.html](https://proceedings.neurips.cc/paper_files/paper/2019/hash/f63f65b503e22cb970527f23c9ad7db1-Abstract.html)
- Roux, J. L., Wisdom, S., Erdogan, H., & Hershey, J. R. (2019). SDR – Half-baked or Well Done? *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 626–630. <https://doi.org/10.1109/ICASSP.2019.8683855>
- Rumberg, L., Gebauer, C., Ehlert, H., Wallbaum, M., Bornholz, L., Ostermann, J., & Lüdtke, U. (2022). kidsTALC: A corpus of 3-to 11-year-old german children's connected natural speech. *INTERSPEECH*, 5160–5164.
- Salazar, J., Kirchhoff, K., & Huang, Z. (2019, January 22). *Self-Attention Networks for Connectionist Temporal Classification in Speech Recognition*. arXiv.org. <https://doi.org/10.1109/ICASSP.2019.8682539>
- Selinker, L. (1972). Interlanguage.
- Shahnawazuddin, S., Adiga, N., Kumar, K., Poddar, A., & Ahmad, W. (2020). Voice Conversion Based Data Augmentation to Improve Children's Speech Recognition in Limited Data Scenario, 4382–4386. <https://doi.org/10.21437/Interspeech.2020-1112>
- Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerry-Ryan, R. J., Saorous, R. A., Agiomyrgiannakis, Y., & Wu, Y. (2018, February 16). *Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions*. arXiv: 1712.05884 [cs]. <https://doi.org/10.48550/arXiv.1712.05884>

- Shen, K., Ju, Z., Tan, X., Liu, Y., Leng, Y., He, L., Qin, T., Zhao, S., & Bian, J. (2023, May 30). *NaturalSpeech 2: Latent Diffusion Models are Natural and Zero-Shot Speech and Singing Synthesizers*. arXiv: 2304.09116 [eess]. <https://doi.org/10.48550/arXiv.2304.09116>
- Shivakumar, P. G., & Narayanan, S. (2022). End-to-end neural systems for automatic children speech recognition: An empirical study. *Computer Speech & Language*, 72, 101289.
- Shobaki, K., Hosom, J.-P., & Cole, R. A. (2000). The OGI kids<sup>2</sup> speech corpus and recognizers. *Proc. 6th International Conference on Spoken Language Processing (ICSLP 2000)*, vol. 4, 258–261. <https://doi.org/10.21437/ICSLP.2000-800>
- Sisman, B., Yamagishi, J., King, S., & Li, H. (2021). An Overview of Voice Conversion and Its Challenges: From Statistical Modeling to Deep Learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29, 132–157. <https://doi.org/10.1109/TASLP.2020.3038524>
- Southwell, R., Ward, W., Trinh, V. A., Clevenger, C., Clevenger, C., Watts, E., Reitman, J., D'Mello, S., & Whitehill, J. (2024). Automatic Speech Recognition Tuned for Child Speech in the Classroom. *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 12291–12295. <https://doi.org/10.1109/ICASSP48485.2024.10447428>
- Su, H., Hu, T.-Y., Koppula, H. S., Vemulapalli, R., Chang, J.-H. R., Yang, K., Mantena, G. V., & Tuzel, O. (2023, September 18). *Corpus Synthesis for Zero-shot ASR domain Adaptation using Large Language Models*. arXiv: 2309.10707 [eess]. <https://doi.org/10.48550/arXiv.2309.10707>
- Sutskever, I. (2014). *Sequence to sequence learning with neural networks*.
- Taal, C. H., Hendriks, R. C., Heusdens, R., & Jensen, J. (2010). A short-time objective intelligibility measure for time-frequency weighted noisy speech. *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 4214–4217. <https://doi.org/10.1109/ICASSP.2010.5495701>
- Tan, X., Chen, J., Liu, H., Cong, J., Zhang, C., Liu, Y., Wang, X., Leng, Y., Yi, Y., He, L., Zhao, S., Qin, T., Soong, F., & Liu, T.-Y. (2024). NaturalSpeech: End-to-End Text-to-Speech Synthesis With Human-Level Quality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(6), 4234–4245. <https://doi.org/10.1109/TPAMI.2024.3356232>
- Tokuda, K., Kobayashi, T., & Imai, S. (1995). Speech parameter generation from HMM using dynamic features. *1995 International Conference on Acoustics, Speech, and Signal Processing*, 1, 660–663 vol.1. <https://doi.org/10.1109/ICASSP.1995.479684>
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). *Llama: Open and efficient foundation language models*.
- Ueno, S., Inaguma, H., Mimura, M., & Kawahara, T. (2018). Acoustic-to-word attention-based model complemented with character-level CTC-based model. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5804–5808.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. *Advances in Neural Information Processing Systems*, 30.
- Wang, C., Rivière, M., Lee, A., Wu, A., Talnikar, C., Haziza, D., Williamson, M., Pino, J., & Dupoux, E. (2021, July 27). *VoxPopuli: A Large-Scale Multilingual Speech Corpus for Representation Learning, Semi-Supervised Learning and Interpretation*. arXiv: 2101.00390 [cs, eess]. <https://doi.org/10.48550/arXiv.2101.00390>
- Wang, D., Wang, X., & Lv, S. (2019). An Overview of End-to-End Automatic Speech Recognition. *Symmetry*, 11(8), 1018. <https://doi.org/10.3390/sym11081018>
- Wang, W., Zhou, Z., Lu, Y., Wang, H., Du, C., & Qian, Y. (2021). Towards Data Selection on TTS Data for Children's Speech Recognition. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6888–6892. <https://doi.org/10.1109/ICASSP39728.2021.9413930>

- Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q., Agiomyrgiannakis, Y., Clark, R., & Saurous, R. A. (2017, April 6). *Tacotron: Towards End-to-End Speech Synthesis*. arXiv: 1703.10135 [cs]. <https://doi.org/10.48550/arXiv.1703.10135>
- Ward, W., Cole, R., & Pradhan, S. (2019). My science tutor and the myst corpus. *Boulder Learn. Inc.*
- Woo, S., Debnath, S., Hu, R., Chen, X., Liu, Z., Kweon, I. S., & Xie, S. (2023). Convnext v2: Co-designing and scaling convnets with masked autoencoders. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16133–16142.
- Yu, D., & Deng, L. (2015). *Automatic Speech Recognition: A Deep Learning Approach*. Springer. <https://doi.org/10.1007/978-1-4471-5779-3>
- Zen, H., & Sak, H. (2015). Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4470–4474. <https://doi.org/10.1109/ICASSP.2015.7178816>
- Zen, H., Senior, A., & Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 7962–7966. <https://doi.org/10.1109/ICASSP.2013.6639215>
- Zen, H., Tokuda, K., & Black, A. W. (2009). Statistical parametric speech synthesis. *speech communication*, 51(11), 1039–1064.
- Zhang, Y., Yue, Z., Patel, T., & Scharenborg, O. (2024). Improving child speech recognition with augmented child-like speech, 5183–5187. <https://doi.org/10.21437/Interspeech.2024-485>
- Zhao, S., Singh, M., Woubie, A., & Karhila, R. (2023). Data augmentation for children ASR and child-adult speaker classification using voice conversion methods, 4593–4597. <https://doi.org/10.21437/Interspeech.2023-702>
- Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., ... Wen, J.-R. (2024, October 13). *A Survey of Large Language Models*. arXiv: 2303.18223 [cs]. <https://doi.org/10.48550/arXiv.2303.18223>