THE UNIVERSITY *of* EDINBURGH

Controlling text-to-speech pronunciation using limited linguistic resources

Jason Fong

Doctor of Philosophy
School of Informatics
University of Edinburgh

2024

Abstract

Correct pronunciation is essential for high-quality text-to-speech (TTS) systems. To achieve this, the majority of TTS systems rely on phonemes as an intermediate representation between input graphemes and output speech. Phonemes are generated by a pronunciation lexicon and grapheme-to-phoneme model. Both of these resources, however, are costly to create and consequently are only available for a limited number of well-resourced languages.

In recent years, end-to-end TTS models, which can be learned entirely from scratch in a data-driven manner, have become more prevalent because they remove the need for creating and maintaining heuristically engineered modules. Although they are often used with phoneme inputs, another advantage of such models is that they enable grapheme-input TTS. While this is a promising step forward for creating TTS voices for under-resourced languages, a significant drawback is the resulting compromise in maintaining precise control over pronunciation.

The primary focus of this thesis is to address these limitations by developing novel methods that enable pronunciation control for grapheme-input TTS systems, without depending on a large phoneme lexicon. I present two main contributions: Firstly, I demonstrate that pronunciation control can be achieved using small phoneme-based pronunciation lexica. Secondly, I demonstrate that ground-truth speech exemplars of word pronunciations can be used to directly control the pronunciations of a TTS system, or to retrieve novel spellings that are correctly pronounced.

The methods presented in this thesis hold the potential to revolutionise the landscape of TTS system development. Through a reduction in the time and expenses involved in creating and maintaining pronunciation resources, this research paves the way for the implementation of high-quality TTS in languages that lack extensive pronunciation resources. Furthermore, these methodologies empower language communities by removing the necessity for linguistic expertise and enabling crowd participation, thereby advancing the universal accessibility of speech technologies across diverse languages worldwide.

Acknowledgements

I would first like to express my deepest gratitude to my primary supervisor, Professor Simon King. Throughout my PhD journey, he has encouraged me to explore my own ideas while offering invaluable guidance and support. His belief in my potential has allowed me to develop a set of skills that will undoubtedly serve me well in the future. I am truly grateful for his mentorship and for fostering an environment where intellectual curiosity could flourish.

I would also like to thank my co-supervisors, Professor Junichi Yamagishi and Dr. Hao Tang. Their generosity with their time and their thoughtful care in guiding my research have been crucial in shaping the direction of my work. Their feedback has refined my ideas and contributed greatly to the quality of this thesis. I deeply appreciate their commitment to helping me grow as a researcher.

I am incredibly thankful to Edinburgh and the Centre for Speech Technology Research for providing not only an excellent academic environment but also a vibrant and supportive community. The connections I have formed here, during moments of laughter, shared struggles through tough times, or over lunch on the third-floor sofas, have been a source of strength and joy throughout my time at the university. The sense of camaraderie within this group is something I will always cherish, and I am hopeful that the connections I've made will last a lifetime.

I would like to express my heartfelt thanks to all my friends and loved ones. You have been by my side through the highs and lows of this journey, making these years not only enriching but also deeply meaningful. Whether it was board game nights, late-night conversations, pot luck dinners, or much needed distractions, your companionship has been an essential part of my experience. The memories we have created together are ones that I will always hold dear. You have helped make Edinburgh feel like home.

Finally, I am profoundly grateful to my parents, whose unwavering love and encouragement over the years have been the foundation of everything I have achieved. Without them, this journey would not have been possible.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

— Jason Fong

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The future is already here, it's just not very evenly distributed.

(William Gibson)

Text-to-speech (TTS) is a crucial component of speech-based interfaces across a wide variety of use cases such as mobile devices, automobiles, and conversational artificial intelligence. TTS converts written text into speech, and, when its output is highly natural and intelligible, enhances user accessibility and engagement. However, the development of TTS systems that can accurately pronounce words across diverse languages, especially those with limited linguistic resources, stands as a significant technical challenge that has not been adequately addressed in existing research. To address this issue, this thesis explores novel methods for specifying pronunciation in a low-resource manner for TTS systems.

In recent years, the synthetic output of TTS systems, particularly those employing the neural paradigm, has considerably improved. Often, their synthetic output crosses the high bar of being indistinguishable from natural human speech. Nevertheless, the production deployment of these systems heavily relies on linguistic resources, such as phoneme-based lexicons. While this approach ensures accurate pronunciations, it poses a significant drawback in terms of global access to TTS technology. The commercial viability of TTS for languages with fewer speakers is often limited, leading to reduced investments in their linguistic resources. Moreover, the absence of trained linguists can further add to the challenge of building TTS for these languages. As a result, TTS practitioners for these languages are often marginalised, lacking the necessary support and expertise to develop

linguistic resources and technologies. Furthermore, there have not been many research efforts that have focused on improving this state of affairs within either the academic, or commercial communities.

In light of these issues, this thesis aims to address the challenges associated with linguistic resource dependence in TTS systems. By exploring both new and neglected research avenues, I provide innovative solutions that do not rely on linguistic resources. Specifically, our work investigates the feasibility of achieving pronunciation control in TTS systems while minimising or even entirely circumventing the use of traditional linguistic resources. Our approaches unlock new possibilities for enhancing the inclusivity of TTS technology.

In this thesis, I introduce several techniques. These include the utilisation of compact, focused phoneme-based lexicons, a departure from the traditional extensive lexicon approach. Additionally, the research explores the integration of ground-truth speech exemplars as a novel means of specifying pronunciation, alongside a novel spelling retrieval method. These techniques represent a significant shift from conventional practices, offering a more resource-efficient and less expertise-dependent route to achieving high-quality pronunciation in TTS systems.

The methodologies proposed in this thesis have the potential to catalyse a significant change in the development of TTS systems. By reducing the dependency on extensive linguistic resources, this thesis paves the way for more equitable language representation in TTS technologies. The broader impact of this work touches upon social and cultural dimensions by enhancing access to TTS by speakers of under-resourced languages and contributing to the preservation of linguistic diversity. Moreover, it democratises the control of these technologies, shifting the balance from large corporations to the language communities themselves.

In conclusion, this thesis details a comprehensive effort to address a critical gap in TTS technology.

1.1 Research goal and contributions

The core question underlying my PhD research is:

Can precise control over the pronunciation of a TTS system be achieved when using few or no linguistic resources such as phoneme transcriptions?

To tackle this broad question, I broke my research down into three stages, reported in six research papers: **1)** verifying the viability of character-input TTS, **2)** searching for a low-resource pronunciation control paradigm, and **3)** using spoken exemplars to control pronunciation. Figure 1.1 is a visual map of each stage of my research, and, below, I give an overview of the contributions of each stage:

Stage 1: Investigating the viability of character-input TTS. I first verified the viability of character-input TTS, finding that the paradigm is robust to word-level transcription errors and also that it performs similarly to a comparable phoneme-input TTS system. I learned that the implicit pronunciation model learned in character-input TTS generalises reasonably well to out-of-vocabulary wordtypes. This was especially noteworthy considering that the TTS model was trained using a dataset that contains far fewer wordtypes than the pronunciation lexicon used to phonemicise the phoneme-input TTS system's inputs. However, when a character-input TTS system makes pronunciation errors, there is no principled or reliable way of correcting them.

Stage 2: Searching for a low-resource method of pronunciation control. I then searched for a low-resource paradigm for pronunciation control. I first investigated the extent to which reducing the size of a phoneme-based pronunciation lexicon impacts the implicit phoneme-to-speech model learned within an end-to-end TTS system. I found that pronunciation robustness is indeed impacted. Consequently, I investigated whether sequences of discrete speech representations could be used to control pronunciations. I found that when spliced into different linguistic contexts, they affect pronunciation when directly conditioning a neural vocoder. Furthermore, I found that this splicing heavily impacted pronunciation robustness, especially when linguistic contexts mismatched. This finding prompted our third and final research stage, which aimed to discover a method to incorporate speech-based pronunciations in a less context-sensitive manner.

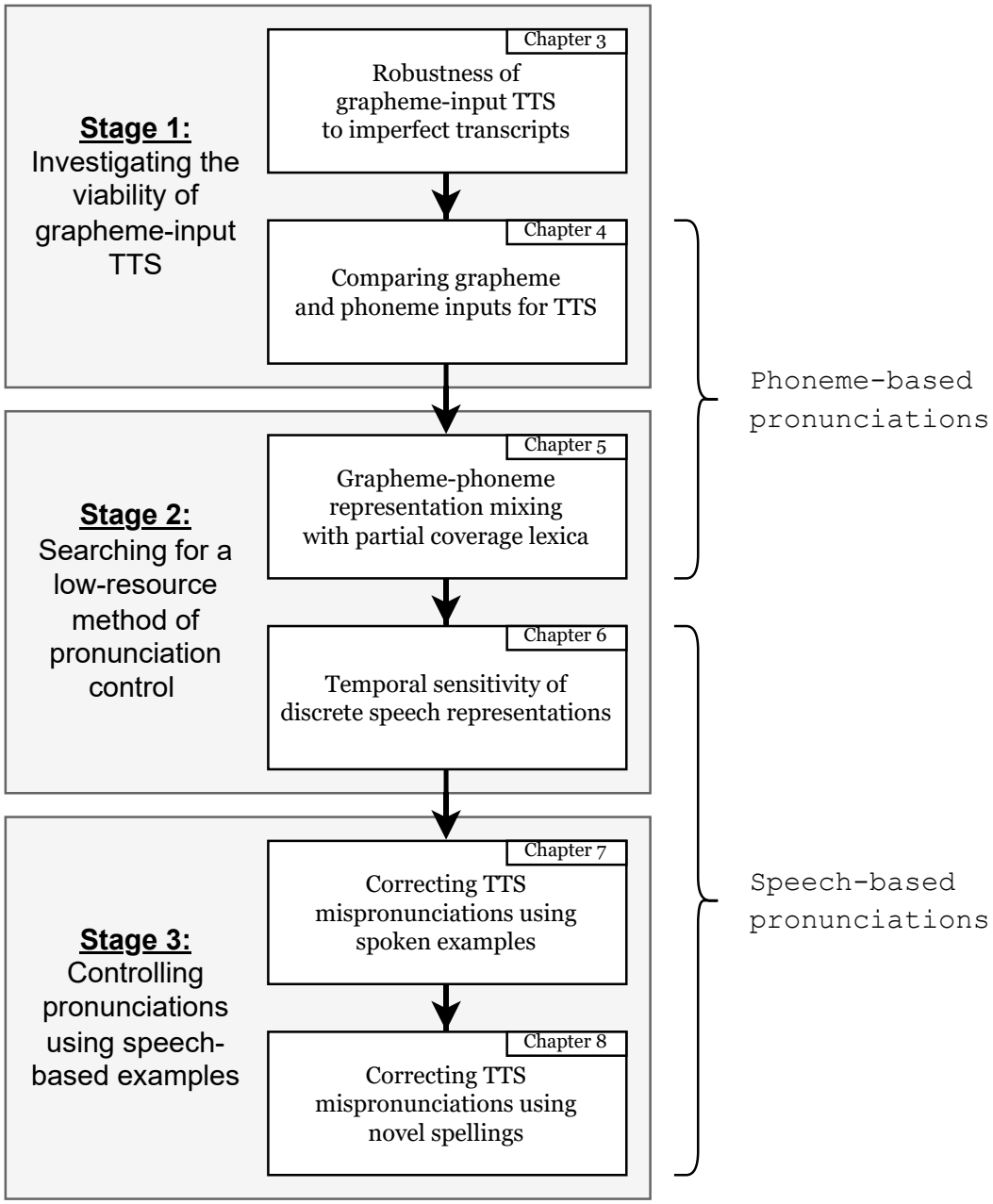

Figure 1.1: Overview of the research projects described in this thesis. On the right hand side of the diagram we group the chapters based on whether they use phoneme or speech to specify pronunciations.

Stage 3: Controlling pronunciations using spoken exemplars. Finally, I focused on developing novel approaches that use spoken examples to control TTS pronunciations. I first directly used discrete speech representations as *input* to a character-input TTS model, and found that this was successful in controlling pronunciation, and was reasonably robust. Secondly I investigated using speech examples to acoustically discover spellings that are pronounced correctly by the TTS system. Such spellings are simpler for untrained persons to understand and worked well in controlling pronunciation.

1.1.1 Additional contributions

In the pursuit of my PhD research, I published 6 papers, which are expanded further in Chapters 3 to 8, to the Interspeech conference and Speech Synthesis Workshop:

- **Investigating the Robustness of Sequence-to-Sequence Text-to-Speech Models to Imperfectly-Transcribed Training Data**, with Pilar Oplustil Gallegos, Zack Hodari, and Simon King (Fong et al., 2019b). Proc. Interspeech 2019.
- **A comparison between letters and phones as input to sequence-to-sequence models for speech synthesis**, with Jason Taylor, Korin Richmond, and Simon King (Fong et al., 2019a). 10th ISCA Speech Synthesis Workshop 2019.
- **Testing the Limits of Representation Mixing for Pronunciation Correction in End-to-End Speech Synthesis**, with Jason Taylor, and Simon King (Fong et al., 2020). Proc. Interspeech 2020.
- **Analysing Temporal Sensitivity of VQ-VAE Sub-Phone Codebooks**, with Jennifer Williams, and Simon King (Fong et al., a). 11th ISCA Speech Synthesis Workshop 2021.
- **Speech Audio Corrector: using speech from non-target speakers for one-off correction of mispronunciations in grapheme-input text-to-speech**, with Daniel Lyth, Gustav Eje Henter, Hao Tang, and Simon King (Fong et al., 2022a). Proc. Interspeech 2022.
- **Spell4TTS: Acoustically-informed spellings for improving text-to-**

speech pronunciations, with Hao Tang, and Simon King (Fong et al., 2023). 12th ISCA Speech Synthesis Workshop 2023.

Aside from the main research undertaken towards my PhD, I have also contributed to several papers with members from my group at the University of Edinburgh, The Centre for Speech Technology Research (CSTR):

- **Where do the improvements come from in sequence-to-sequence neural TTS?** with Oliver Watts, Gustav Eje Henter, and Cassia Valentini-Botinhao (Watts et al., 2019). 10th ISCA Speech Synthesis Workshop 2019.
- **Exploring disentanglement with multilingual and monolingual VQ-VAE**, with Jennifer Williams, Erica Cooper, and Junichi Yamagishi (Williams et al., 2021). 11th ISCA Speech Synthesis Workshop 2021.

Furthermore, I wrote two papers, both related to TTS, during my two internships at Meta:

- **Improving Polyglot Speech Synthesis through Multi-task and Adversarial Learning**, with Jilong Wu, Prabhav Agrawal, Andrew Gibiansky, Thilo Koehler, and Qing He (Fong et al., b). 11th ISCA Speech Synthesis Workshop.
- **Towards zero-shot Text-based voice editing using acoustic context conditioning, utterance embeddings, and reference encoders**, with Yun Wang, Prabhav Agrawal, Vimal Manohar, Jilong Wu, Thilo Köhler, Qing He (Fong et al., 2022b). arXiv preprint.

Finally, throughout my time studying in the University of Edinburgh, I have been a lab demonstrator and coursework marker for the Speech Processing and Speech Synthesis courses.

Chapter 2

Background

In this section, I describe the scientific background to the thesis, and the methods I employed to undertake the research. The background covers various topics in Linguistics, Machine Learning, and Speech Technology.

2.1 Low-resource languages

In this thesis, we define **low-resource languages** as those for which there is a limited availability of annotated linguistic data, trained linguists, or speech technology specialists. These languages often remain underrepresented in the fields of computational linguistics and speech technology, due to the absence of a compelling commercial incentive for investment. Consequently, despite the world's linguistic diversity, encompassing over 7,000 languages, a significant majority of technological resources for TTS systems are focused on a small subset of high-resource and commercially viable languages.

However, the inclusion of low-resource languages when designing speech technology systems is important for several reasons. First, from a linguistic perspective, these languages often embody unique phonetic, syntactic, and semantic features that can enrich our understanding of human language. Consequently, integrating low-resource languages leads to the development of more robust and versatile TTS systems, capable of generalising to a wider variety of linguistic phenomena. However, developing TTS systems for low-resource languages faces numerous challenges. The scarcity of data and lack of linguistic annotations can result

in inferior quality when training TTS systems. For example, Latorre et al. (2019b) show that training a neural TTS system on a relatively large amount of single speaker data (approx. 25,000 utterances) can still result in unstable synthesis outputs. While stability *can* be improved by including training data from additional speakers this increases cost.

Recent advances in TTS technology, however, offer promising solutions for low-resource languages. Techniques such as transfer learning (Tu et al., 2019), where a model trained on several high-resource languages is adapted to a low-resource language, have shown potential. Another approach is the use of features from self-supervised models (Kim et al., 2022) which can be adopted within TTS systems so that less annotated data is required. The use of such methods is crucial for creating TTS systems that are inclusive to a diverse global population.

The inclusion of low-resource languages in TTS systems has significant implications for global communication and accessibility. It enables speakers of these languages to access technology in their native tongue, fostering inclusivity and reducing the digital divide. Moreover, it aids in the preservation and documentation of languages that are at risk of disappearing, contributing to cultural and linguistic diversity.

2.2 Pronunciation

Pronunciation refers to the manner in which a word or language is spoken. It's important to recognise that while there is no universally *correct* mode of speaking, commonly accepted pronunciations exist within linguistic communities. For instance, the pronunciation of the word 'niche' as 'nitch' (/ˈnɪt͡ʃ/) is prevalent in the United States but less so in the United Kingdom. In this context, this pronunciation variant is deemed correct in American English but less standard in British English. This thesis defines these widely accepted forms as 'correct pronunciations' and contrasts them with 'mispronunciations,' which can lead to communication barriers.

It is also important to note that the acoustic features defining correct or incorrect pronunciation vary across languages. In languages like Chinese or Thai, tone is a distinguishing feature, making the accurate perception and production of tone essential for understanding and effective communication. Moreover, although

pronunciation is frequently influenced by a language's orthography, it is distinct from the act of reading.

In this thesis, we employ a definition of pronunciation which is more specific compared to the broader scope found in much of the existing linguistics literature. We define pronunciation as the phonemic sequences that constitute and contrast a word within a given language. This definition excludes various other elements of speech acoustics, including prosody, syllabic stress, syllable segmentation, speaker-specific variations, accents, rhythmic patterns, and sociolinguistic influences. It is recognised that certain additional aspects are crucial for lexical differentiation in some languages. For instance, lexical stress, tones, and quantity can influence pronunciation, providing essential phonological contrasts necessary for lexical access. This particular thesis however deliberately omits the examination of non-phonemic pronunciation features to narrow the focus of our experiments.

2.2.1 Factors influencing pronunciation

Pronunciation is subject to a myriad of influences, including language learning duration, exposure to other languages, speech disorders, geographic location, ethnicity, social class, and educational background. An example is the phenomenon of 'th'-fronting in British English (Stuart-Smith and Timmins, 2006), where the letters 'th' in a word such as '**th**ree' can be pronounced as [f] instead of [θ]. While generally intelligible, this variation can occasionally cause confusion.

In the design of TTS systems, acknowledging and incorporating such pronunciation variations is vital. This ensures that the synthesised speech not only reflects the diversity of spoken language but also enhances the intelligibility and accessibility of TTS technology. By capturing these nuances in pronunciation, TTS systems can become more inclusive and cater to a broader spectrum of users.

2.2.2 Pronunciation errors

Pronunciation errors in language use can take various forms, including confusing similar sounding words, forgetting a word's pronunciation, or making errors during rapid speech. Errors also occur when reading aloud, as detailed in Section 2.3. Since mispronunciations can lead to misunderstandings and negative perceptions, particularly for non-native speakers, accurate pronunciation is often a concern. (E)

More generally, while maintaining intelligibility, the quality of pronunciation can only be lowered in contexts where meaning can still be disambiguated. Lindblom's theory of Hyper and Hypo-articulation suggests that speakers adjust how clearly they speak based on the listener's needs. They speak more clearly (hyper-articulate) when the listener needs more information and reduce clarity (hypo-articulate) when the listener can use other clues. To ensure speech remains understandable, speakers only reduce clarity if the listener can still distinguish the words from similar-sounding ones Lindblom (1990). Pronunciation improvement typically occurs through deliberate practice and feedback.

In the context of TTS, the presence of pronunciation errors in training data can compromise the quality of TTS models, leading to less accurate and natural-sounding speech output. Moreover, when TTS systems produce pronunciation errors, it significantly reduces their utility and acceptability, especially in contexts where precise communication is critical. Thus, ensuring high-quality training data and incorporating mechanisms to detect and correct pronunciation errors are essential for the development of reliable and effective TTS technology.

2.2.3 Pronunciation learning

Pronunciation learning encompasses both passive and active methods. Passively, individuals acquire pronunciation through media consumption and conversation. Actively, they engage in learning through explicit feedback from educators or language materials. A fundamental aspect of this process is understanding a language's phonetic inventory, which enables learners to recognise and subsequently correct their pronunciations upon exposure to the correct forms. Non-verbal feedback, such as listeners' expressions of confusion, also plays a role in signalling potential pronunciation issues, contributing to the learning process.

The strategies for pronunciation acquisition are significantly influenced by one's familiarity with the target language and the amount of exposure to it. Typically, native speakers acquire pronunciation through experiential means rather than formalised systems like the International Phonetic Alphabet (IPA) which non-native speakers may prefer. Nevertheless, the emergence of online dictionaries with audio pronunciations has introduced a new dimension to pronunciation learning, offering an accessible resource that can potentially replace traditional methods.

In TTS systems, similar principles can apply: character-input TTS systems 'learn' pronunciation by analysing large quantities of spoken language data, mimicking the passive acquisition observed in humans. However, phoneme-input TTS systems require a more active feedback process; refinement in these systems originates from user feedback, and pronunciation corrections must be manually added to the phoneme-based lexicon. This process is more akin to how learners adjust their pronunciation based on input from educators.

2.3 Reading aloud

Reading, a complex cognitive process, involves deciphering sounds and meaning from visual symbols that represent language. It is used for language acquisition, communication, and idea dissemination. Reading can be a silent, internal activity, with some readers interpreting text without auditory representation, a process known as subvocalisation (Slowiaczek and Clifton Jr, 1980). Alternatively, others may internally sound out words. Reading aloud, whether for personal benefit or for communicating with others, represents a distinct aspect of this skill.

In this work, we do not distinguish between silent reading with subvocalisation and reading aloud. Our focus is on the universal process by which humans convert the orthographic form of a word into its phonetic sequence. This process closely aligns with the challenges encountered in traditional TTS pronunciation. However, insights from the practice of reading aloud could inform innovative TTS pronunciation strategies. These might include leveraging semantic or contextual information or understanding the phonotactics of foreign languages to improve word pronunciation predictions.

2.3.1 Dual route hypothesis

The dual route hypothesis is a central theory in the study of reading aloud (Coltheart, 2005). It posits two cognitive pathways involved in this process: the lexical route and the non-lexical (or sublexical route). This hypothesis has helped researchers explain a range of phenomena related to both typical and atypical reading patterns (Jackson and Coltheart, 2013).

The **lexical** route allows readers to recognise familiar words by sight, retrieving their pronunciations from a mental 'lexicon', akin to a dictionary lookup. This

internal lexicon encompasses all learned words, including those like 'colonel' or 'pint', which do not conform to typical character-to-sound patterns in English. However, the lexical route is, by definition, ineffective for unfamiliar words.

Conversely, the **non-lexical** route involves phonetically decoding words by breaking them down into constituent parts (e.g. letters, graphemes, and phonemes (Rey et al., 2000)) and applying phonological rules to construct a spoken representation. While useful for unfamiliar words, it is prone to errors with foreign words or neologisms because the phonological rules are learnt from common words.

Research has shown that the dual route hypothesis impacts reading speed, particularly with irregular words that defy conventional phonetic rules (Bergmann and Wimmer, 2008). The differing pronunciations from the lexical and non-lexical routes can *conflict*, requiring resolution.

In TTS, the dual route hypothesis informs the design of pronunciation models. The lexical route parallels the use of pre-defined pronunciation dictionaries in TTS, while the non-lexical route is similar to explicit grapheme-to-phoneme (G2P) conversion. Understanding these routes aids in creating TTS systems that can effectively handle a wide range of words, including irregular and novel terms, thus enhancing the naturalness and accuracy of synthesised speech.

In this thesis, we use the terms *graphemes* and *characters* interchangeably when referring to individual letters or combinations of letters in English, as they often function similarly by representing specific sounds or units of meaning. However, we recognise that in other languages, this relationship varies. For instance, in Japanese, *Kana* characters often correspond to syllables or moras, while in Chinese, *Hanzi* characters typically represent morphemes, with each character conveying a distinct meaning that may correspond to a full word or part of a word.

2.3.2 Grapheme-to-phoneme correspondence

Grapheme-to-phoneme correspondence is the systematic relationship between written symbols (graphemes) and their associated spoken sounds (phonemes) in a language. A language is considered orthographically **transparent** when it adheres closely to consistent grapheme-to-phoneme mappings with few exceptions. For instance, Spanish, with its highly consistent mappings, is considered transparent, whereas English, having evolved extensively and incorporating numerous loanwords,

is less so. This disparity partially explains why English is more challenging for L2 learners, due to its **opaque** orthography. Consequently, Spanish speakers with reading disorders like dyslexia experience less impairment compared to English speakers (Ziegler and Goswami, 2005), as they can rely more on consistent phonological rules rather than memorising exceptions.

In TTS systems, grapheme-to-phoneme correspondence plays a pivotal role. For languages with high orthographic transparency, like Spanish, TTS systems can apply straightforward phonetic rules for accurate pronunciation. In contrast, languages with opaque orthography, like English, require a more sophisticated pipeline capable of handling numerous exceptions and irregularities. This difference significantly influences the complexity and effectiveness of TTS systems across various languages.

2.3.3 Low-resource languages with opaque orthography

While low-resource languages with opaque orthography present ideal test cases for our technology, pronunciation correction is universally beneficial given that TTS systems are not infallible and will inevitably make errors. Below, in Table 2.1, we list several such languages that are both low-resource and exhibit opaque orthographic systems.

Table 2.1: Examples of low-resource languages with opaque orthography

Language	References
Tibetan	Beyer (1992); Tournadre (2014); Wang et al. (2023)
Persian	Baluch (2013); Nayernia et al. (2019)
Faroese	Árnason (2011); Barnes and Weyhe (2013)
Thai	Winskel and Iemwanthong (2010); Vibulpatanavong and Evans (2019)

2.4 Phonetics and phonology

2.4.1 Describing speech sounds

Linguists have developed various systems to describe speech sounds in a language-independent manner. This is necessary for transcription and so that different languages can be compared with each other. These descriptions are often discrete,

categorising articulatory features that are inherently continuous. Common discrete divisions in the International Phonetic Alphabet (IPA) include consonants versus vowels, place and manner of articulation for consonants, and height, backness, and rounding for vowels.

In text-to-speech (TTS) systems, accurately describing and replicating speech sounds is crucial. The discrete categorisations used in phonetics have provided a foundation for TTS systems to synthesise speech. The understanding and implementation of these linguistic categorisations, has enabled TTS practitioners to build systems that can produce intelligible speech outputs. More recently, advancements in neural models have also enabled generating outputs that closely resemble human speech from discrete phonetic symbols.

2.4.2 Phones

A phone is any distinct sound produced by speakers, which may or may not be important to the meaning of words. Phones are perceived as discrete categories (e.g., [ə] vs. [i]), but when analysing large phonetic inventories, the formant frequencies (F1 and F2) show continuous overlap rather than distinct clusters. This phenomenon, known as the 'perceptual magnet effect' (Kuhl, 1991), indicates that perception of phone categories changes discretely when moving through the F1/F2 space. This effect can cause a phenomena named phonological deafness, for instance, Japanese learners of English often struggle to distinguish [r] and [l] sounds due to the absence of these distinct phones in Japanese, which only has the liquid phoneme /r/, realised usually as an apico-alveolar tap [ɾ], leading to difficulties in perception and reproduction.

However, in TTS systems, phones are not typically used to represent pronunciations. This is primarily because phones include a vast array of subtle variations that are context-dependent and speaker-specific. Representing all these variations in a TTS system would be extremely complex and possibly computationally demanding. Instead, TTS systems generally use allophones, which are more abstract, providing a more consistent and efficient way to represent the sounds of speech that are relevant to generating spoken language across diverse conditions and speakers.

2.4.3 Phonemes

A phoneme is the smallest unit of sound in a language that serves to convey meaning, distinguishing one word from another (Twaddell, 1935). It is an abstract concept representing a set of speech sounds that are perceived as a single distinctive sound in a language. Phonemes are not the sounds themselves, but are cognitive categories or classes of sounds that speakers recognise as equivalent.

For example, consider the phoneme /p/ in English. It can be heard in words like "**p**at", "s**p**at", and "ta**p**". Although the actual sound of /p/ varies slightly in each of these words due to its position and surrounding sounds (these subtle variations of /p/ are called allophones), it is still perceived by English speakers as the same underlying sound, or phoneme /p/. This phoneme is crucial for meaning; changing it to /b/ in "pat" to form "bat" changes the meaning of the word, demonstrating the phoneme /p/'s role as a meaningful unit in the language. This characteristic of changing meaning with a change in sound is what defines /p/ as a phoneme in English, and is also what defines "pat" and "bat" as a minimal pair (i.e. a pair of words that differ in only one phoneme).

In TTS, allophones have traditionally formed the backbone of pronunciation modelling. By abstractly representing speech sounds, TTS systems can efficiently process and synthesise speech, ensuring that the produced speech varies meaningfully with different phonemes. This phonemic foundation is crucial for creating TTS outputs that are both intelligible and linguistically accurate.

2.5 Neural networks

Neural networks are a fundamental component of modern machine learning models. Neural networks were originally inspired by the structure and function of the human brain using layers of interconnected nodes, or "neurons" (McCulloch and Pitts, 1943; Rosenblatt, 1958). Each neuron receives input, processes it through a mathematical function, and passes the output to subsequent neurons. The network has an input layer to receive the data, hidden layers to process the data, and an output layer to produce the final result. Neural networks learn to perform tasks by adjusting the weights of connections between neurons based on the inputs they receive, through an optimisation process called backpropagation (Rumelhart et al., 1986), which adjusts these weights to minimise the difference between the

network's output and the desired output (often referred to as the network's "loss").

The power of neural networks is underscored by the Universal Approximation Theorem (Cybenko, 1989). This theorem states that a feed-forward network with just one hidden layer and non-linear activation functions can approximate *any* continuous function. This theorem highlights the capability of neural networks to model complex and diverse functions, contributing to their widespread adoption in various machine learning tasks in recent years. Furthermore, a critical advantage of neural networks is their ability to automatically learn higher-level features, thereby eliminating the need for manual feature engineering. This is primarily enabled by the architecture of neural networks, which often consist of multiple layers. Each layer can learn increasingly complex features from the data it receives (Bengio et al., 2013). Consequently, neural networks can efficiently handle the variability and complexity inherent in the data from various domains such as vision, natural language, and human behaviour. Automatic feature representation has been crucial in enabling predictive or generative AI systems that perform tasks equal to or even better than human level.

The integration of neural network models has been particularly beneficial to the field of text-to-speech (TTS). Their ability to automatically learn and adapt to complex patterns in speech has enabled the generation of more natural and human-like synthetic speech. Traditional TTS systems relied heavily on handcrafted features and extensive linguistic expertise, but neural networks can automatically learn similar intermediate features from large datasets of speech. Additionally, neural networks' flexibility allows TTS systems to be easily tailored to specific voices or accents, further enhancing their applicability.

In the rest of this section, we will give a description of specific neural network architectures relevant to TTS.

2.5.1 Feed-forward neural networks

Feed-forward neural networks are the simplest type of neural network architecture. In these networks, information moves in only one direction, from the input nodes through to the output nodes.

In modern TTS systems, feed-forward neural networks are rarely used as standalone architectures due to their inability to handle sequential and contextual input data.

This means that they lack the ability to model long-range temporal dependencies and the dynamic variations within speech, such as intonation, stress, and rhythm, which are essential for generating natural-sounding speech.

2.5.1.1 Convolutional neural networks (CNN)

Convolutional Neural Networks (CNNs) are a specialised type of neural network that excel in processing data that exhibit hierarchical features, by using efficient convolution operations. Figure 2.1 visualises the typical architecture of a CNN based image classification model. It mainly consists of a feature extractor, composed of convolutional layers, and is followed by fully connected layers that use aggregated information from the extracted features to make image classification predictions. CNNs have notably demonstrated state of the art performance in the domain of image classification (Krizhevsky et al., 2012). To perform this task well, CNNs learn to recognise edges, colours and textures at early layers, parts of objects at intermediate layers, and ultimately whole objects or scenes at the topmost layers. Figure 2.2 shows a visualisation of convolutional kernel outputs at the first layer of a trained CNN image classifier, illustrating the edge detection capabilities that are learnt without any prior knowledge.

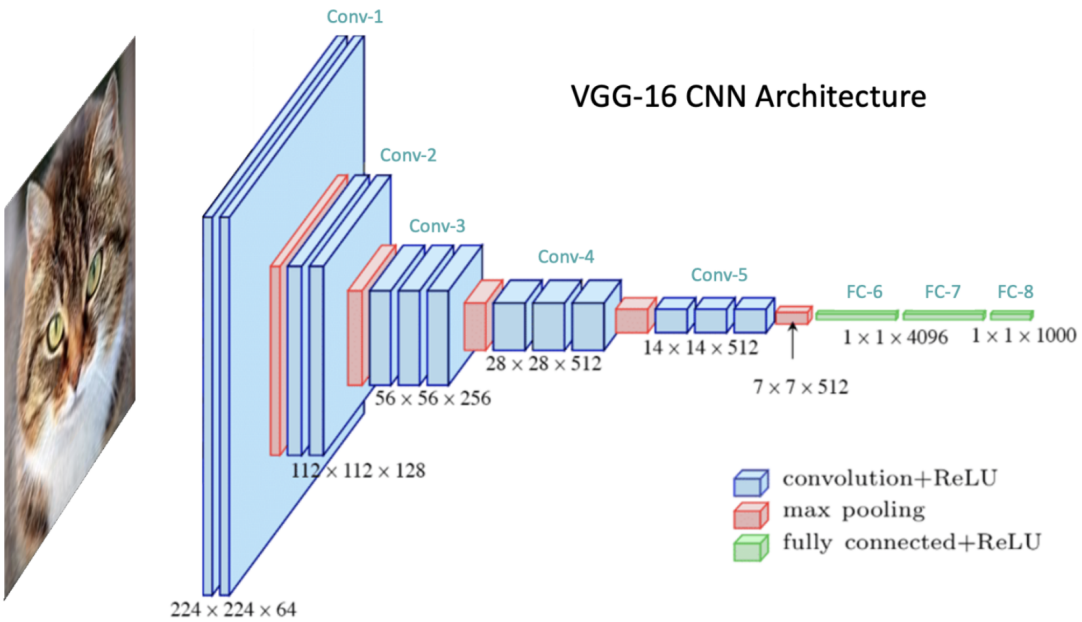

Figure 2.1: An example of the architecture of a typical CNN image classifier. Image credit (Mallick, 2024).

Additionally, CNNs have found significant applications in speech processing,

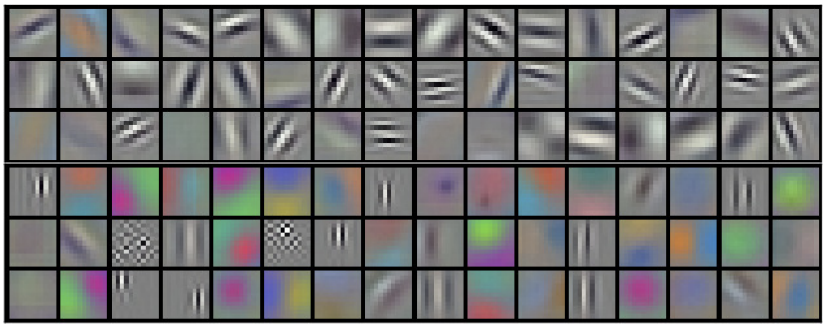

Figure 2.2: Convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. CNNs are able to learn kernels akin to edge detection algorithms automatically and from scratch. Image is taken from the first ImageNet challenge model that used CNNs, rather than more traditional hand engineered features (Krizhevsky et al., 2012).

particularly for feature extraction and acoustic modelling (Ping et al., 2018; Baevski et al., 2020; Hsu et al., 2021). Their capability to detect local patterns within input data renders them highly effective for tasks like identifying phonetic components and distinguishing between speech and non-speech segments. This efficacy originates from the convolutional layers of CNNs, which apply filters to discrete sections of the input, thereby capturing local temporal dependencies and subtle variations in speech frequencies and patterns.

However, akin to feed-forward networks, CNNs are not typically utilised as the sole architecture for the entire TTS pipeline. This is due to their limited capacity to model the long-term temporal dependencies that are inherent in speech. In TTS and Speech Processing systems, CNNs are often combined with other neural network types, such as recurrent neural networks (RNNs) or transformers, which are equipped to manage the contextual nature of speech. This hybrid approach leverages the strengths of CNNs in efficient feature extraction while compensating for their sequence modelling limitations.

2.5.2 Sequence-to-sequence neural networks

Sequence-to-Sequence (Seq2Seq) neural networks are a category of neural networks designed to handle arbitrary length sequences of input and/or output data. Consequently, Seq2Seq models are used in natural language processing, machine translation, and TTS systems. Seq2Seq models have the ability to model the

dependencies between elements in a sequence, making them particularly suitable for tasks that require an understanding of context and the sequential nature of the data. In this subsection we describe two popular Seq2Seq architectures that we used in our research, Recurrent Neural Networks and Transformers.

2.5.2.1 Recurrent neural networks (RNN)

Recurrent Neural Networks (RNNs) are characterised by their ability to maintain a memory of previous inputs through internal states, allowing them to model temporal patterns in the data. Figure 2.3 shows a conceptual overview of RNNs, and how they progressively process their inputs in a sequential fashion. Within RNN-based TTS architectures such as Tacotron (Wang et al., 2017b; Shen et al., 2018a), RNNs are used for modelling temporal dependencies in input or output speech sequences, capturing dynamic speech features such as prosody, intonation, and rhythm. This enables these systems to produce human-like synthetic speech.

However, RNNs face challenges, particularly the vanishing gradient problem (Hochreiter, 1998), which makes them less effective at capturing long-term dependencies in longer sequences. This limitation was partially addressed by advanced RNN variants such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Chung et al., 2014) networks, which introduce mechanisms to better retain information over extended sequences. Nevertheless, the issue of modelling long-term dependencies has since been addressed by the introduction of the Transformer architecture (Vaswani et al., 2017).

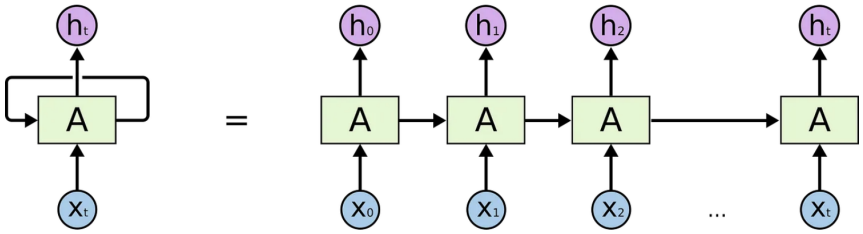

Figure 2.3: High-level illustration of recurrent neural networks (RNNs). A is a recurrent unit such as an LSTM unit or a GRU unit. $x_{0:t}$ are inputs to the model, such as letters, words, or MFCC frames. $h_{0:t}$ are hidden states output by the recurrent unit, that can either be input to another neural network, or used directly as an output. Image credit (Ma, 2024).

2.5.2.2 Transformers

Transformers represent a significant advancement in the field of Seq2Seq models (Vaswani et al., 2017). They use the self-attention mechanism, which allows for direct modelling of dependencies, regardless of their distance in the sequence. Furthermore, unlike RNNs, Transformers are not required to process sequentially, allowing for significantly increased efficiency through parallelisation.

In the domain of TTS, Transformers have been revolutionary, particularly with the introduction of models like Transformer TTS (Vaswani et al., 2017), FastSpeech 2 (Ren et al., 2020), and Fastpitch (Łańcucki, 2021). These architectures leverage the Transformer's ability to model complex long-term dependencies and generate high-quality, natural-sounding speech. The key component of Transformers, the self-attention mechanism, enables the model to weigh the importance of different parts of the input data differently, which is particularly beneficial for modelling the nuances and contextual variations in speech. Figure 2.4 shows how self-attention can dynamically determine the importance of elements in the input sequence relative to a single particular element.

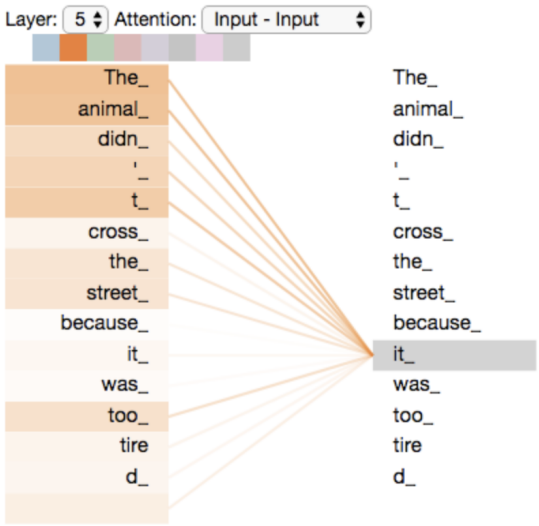

Figure 2.4: Illustration of the self-attention map produced by a transformer encoder processing an English source sentence. The encoder was trained (along with a decoder) to perform machine translation from a source language sentence to a target language one. Image credit (Alammar, 2024).

2.6 Text-to-speech

The goal of Text-to-Speech (TTS) is to convert written text into spoken words in a manner that simulates the naturalness and expressivity of human speech. TTS applications include virtual assistants, audiobooks, and accessibility tools for those with reading difficulties. In this section we describe both traditional and modern paradigms of TTS, and then, in the context of TTS, discuss the importance of pronunciations and how they can be represented and controlled.

2.6.1 Traditional paradigms of text-to-speech

Two traditional paradigms of TTS are concatenative speech synthesis and statistical parametric speech synthesis, which are commonly referred to as **unit selection** and **SPSS** respectively. This subsection provides an overview of how each paradigm functions, explains their reliance on a front-end and pronunciation lexicon to generate linguistic features, and describes how they handle pronunciation correction.

At a high level, unit selection and SPSS can both be considered as *waveform generation* methods or *back-ends*. They generate a speech waveform from an input sequence of linguistic features that typically include phonemes and contextual information that correspond to the desired input text. Such linguistic features are generated by a front-end. Figure 2.5 shows a broad overview of how unit selection and SPSS fit into the TTS pipeline. Phonemes, in comparison to graphemes, map to the acoustics of a language in a strictly monotonic fashion. Thus, phonemes are more easily aligned to speech than graphemes and consequently phoneme-to-speech mappings are more trivially deduced. These advantages of using phonemes as an input representation benefits TTS greatly, as its core machine learning task boils down to predicting speech from contextual input features, which is easier given a good alignment. Furthermore, the alignment between phonemes and speech is easily obtainable through forced alignment via an Automatic Speech Recognition (ASR) system or tool such as Kaldi (Povey et al., 2011) or Montreal Forced Aligner (McAuliffe et al., 2017).

Unit selection generates speech by concatenating short segments of recorded speech that fit well together and match the intended input specification represented as a stream of linguistic features (Hunt and Black, 1996; Black and Taylor, 1997).

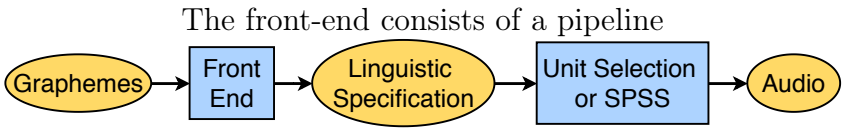

Figure 2.5: High-level overview of the front-end/back-end approach to text-to-speech. Here the back-end, either unit selection or SPSS, generates synthetic speech that corresponds to underlying contextual linguistic features generated by the front-end. The linguistic specification commonly consists of phonemic rather than graphemic units as phonemes more trivially align to acoustics. This allows the use of simpler waveform generation methods such as unit selection or SPSS, which cannot easily deal with non-monotonic alignments between input and output.

SPSS, on the other hand, uses a statistical acoustic model learned from a database of speech to predict acoustic parameters from linguistic features (Tokuda et al., 2000, 2002; Zen et al., 2007). The acoustic parameters are then synthesised by a vocoder into speech (Yoshimura et al., 1999). Both methods generate speech that is intelligible and somewhat natural. Unit selection voices often exhibit audible artefacts at the join between neighbouring segments. This limitation becomes more severe when the system is built from a small speech database. In such a case, there is a limited selection of speech segments, resulting in concatenated segments that may not fit well together or precisely match the intended linguistic specifications. SPSS voices exhibit no such join artifacts, but their speech lacks natural expressivity and intonation due its use of simple regression methods (e.g. regression trees) that map from linguistic features to acoustic parameters without taking into account much contextual knowledge from previous or future timesteps. This lack of contextual modelling applies to both the input linguistic features and speech previously generated by the model.

One key characteristic of traditional TTS paradigms such as unit selection or SPSS is their reliance on a front-end to generate contextual linguistic features for most languages. This reliance is due to their need to use input features (often phonemes) that monotonically align to the acoustic features. A monotonic alignment means that the sequence of input features must correspond directly and sequentially to the acoustic features without any reordering. Consequently, for many languages grapheme inputs cannot be trivially used with traditional TTS paradigms as a monotonic alignment between graphemes and acoustics does not hold. For example, in English there exist words with silent letters, which do not

align to any sounds, for example '**p**neumatic' or '**g**naw'.

2.6.2 Representing pronunciation for text-to-speech

While it is theoretically possible to build a word-based TTS system, in which recordings of each individual word are used, this approach is not practical due to the sheer number of words in a language. Even for a relatively small vocabulary, it would be challenging to acquire sufficient recorded speech to cover all possible word pronunciations, especially when accounting for contextual influences on word pronunciations (such as place of articulation assimilation). This data sparsity problem becomes even more severe for languages with rich agglutinative morphology, such as German, Turkish, and Finnish, where words can be composed of many morphemes.

As a result, TTS systems have used phonemes as a representation of pronunciation. Compared to whole words and graphemes, phonemes offer several advantages. They effectively address the data sparsity issue, reflect word pronunciation more consistently, and exhibit a monotonic alignment with the speech signal. However, despite the advantages of using phonemes, they are not a perfect representation of pronunciation. Phonemes are discrete representations of a continuous acoustic space. Hence, they offer a coarse reflection of the articulatory features and events that occur during speech. For instance, the utterance "ten boys" has the phonemes /ˈtɛn bɔɪz/. However, when spoken quickly, it may sound like /ˈtɛm bɔɪz/ due to the process of place of articulation assimilation (Taylor, 2009, p. 195), where the alveolar place of articulation of the /n/ in the first word "ten" assimilates to the bilabial place of articulation of the /b/ in the second word "boys". This assimilation process is hard to predict, and the rate of speech determines the continuous change from /n/ to /m/, rather than a discrete flip at a specific point. As a result, it can be deemed inappropriate to model the articulatory process of speech discretely. Additionally, phoneme transcription methodology is not well-defined, and different phoneticians may transcribe the same speech in various ways.

While there are many limitations associated with using phonemes for TTS , in practice they remain the most widely adopted form of pronunciation representation. One notable advantage of phonemes is their ability to abstract away from specific variations in pronunciation, focusing instead on the fundamental sound patterns

that convey meaning. By using phonemic pronunciations in the lexicon, the input to TTS systems becomes more consistent, offloading the handling of variability in pronunciation across different speakers and dialects to the acoustic model.

2.6.3 Correcting text-to-speech pronunciations

In this thesis we define a pronunciation correction as successfully controlling the pronunciation of a mispronounced word by specifying its correct pronunciation to the TTS model. Pronunciation correction in TTS systems can be robust, meaning that the generated speech waveform matches the intended input text, if the training data was accurately labelled by the front-end. A simple approach to achieve pronunciation correction is by adding an phonetician's correction for a mispronounced word to the pronunciation lexicon. This approach is often taken by companies that require voices with accurate pronunciations and have the means to invest in a high-quality and large pronunciation lexicon. In the remainder of this subsection, we describe the two common resources that are needed to control TTS pronunciations: the front-end and the pronunciation lexicon. We also describe some of the costs and trade-offs associated with each of these resources.

2.6.3.1 The front-end

The front-end consists of a pipeline of submodules and is responsible for processing input text into a monotonic stream of linguistic features by disambiguating complex grapheme-to-acoustic (G2A) relations (Black and Lenzo, 2000). These linguistic features are then used as input to an acoustic model to generate speech. The richer the contextual information that is analysed and added to the features, the better the features correlate with the output acoustics, resulting in more natural sounding TTS output. In order to improve the quality of the resulting TTS voice, the mapping between linguistic features and speech segments in the training data should be as monotonic and predictable as possible. For example, part of speech tagging is typically performed so that homographs (i.e. words that are written the same but are pronounced differently) can be disambiguated to their correct pronunciation according to the context in which they occur, thus improving both the correctness and alignment of the input features to the audio. Similarly, text normalisation expands numbers and abbreviations to their verbalised form (e.g. from "1984" to "nineteen eighty four".

After front-end processing, the phonemes in the speech data are delimited and forced aligned to the linguistic features, enabling the labelling of input-output pairs for the TTS acoustic model to be built or trained from. The phonemicisation of a word follows two pathways depending on whether it occurs within the pronunciation lexicon or not (see Figure 2.6). In the former case, phoneme sequences of words that exist in the lexicon are used directly. However, for out-of-vocabulary words (OOVs), the phoneme sequences must be predicted using a separate Grapheme-to-Phoneme (G2P) model. Although the phoneme sequences predicted by G2P models can be frequently correct, their accuracy cannot be guaranteed due to the existence of words, such as loan words and novel words, that are consistently added to a language over time. Therefore, the use of G2P models for OOVs remains a challenge for TTS systems.

While the use of front-ends for TTS has had success, there are several major downsides. Firstly, the modules that comprise the pipeline are complex to design and build, requiring significant linguistic data and speech technology expertise, making them expensive to construct. Additionally, these pipelines are challenging to improve because each submodule must be optimised separately, making it difficult to obtain predictable results from the full pipeline. This downside of pipeline front-ends is mitigated with the use of more modern neural end-to-end TTS models, which learn front-end like modelling from scratch with a single neural network based model. These models are described further in Section 2.6.4.

2.6.3.2 Grapheme-to-Phoneme modelling

Grapheme-to-Phoneme (G2P) is the task of predicting phoneme sequences from grapheme sequences.

The letter-to-sound approach is a rule-based method for G2P that constructs rules mapping sounds from letters. These rules are often contextual, meaning that preceding and following letters are part of the rule, thereby improving the accuracy of phoneme prediction. These rules can be manually written (Elovitz et al., 1976) or generated from data using probabilistic methods (Van Den Bosch and Daelemans, 1993; Bagshaw, 1998), and are frequently organised in a binary decision tree structure.

In recent years, neural networks have been increasingly employed for G2P tasks.

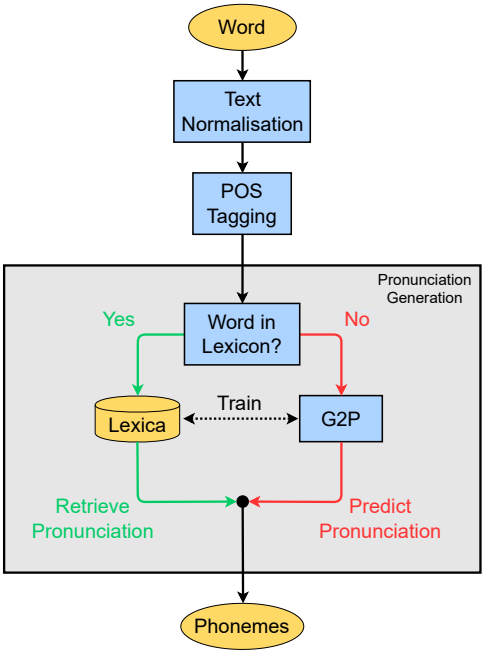

Figure 2.6: Two-pronged approach to pronunciation generation from grapheme inputs. This approach is used in the front-ends of most TTS systems because full phonemicisation of training transcripts has been found to be best when optimising for pronunciation accuracy and robustness of corrections. Typically G2P models are trained using word-pronunciation pairs in the lexicon and then used to predict the pronunciation of OOV words.

These models are capable of end-to-end modeling, using recurrent models such as RNNs (Rao et al., 2015) or autoregressive transformers (Yolchuyeva et al., 2020) to predict phoneme sequences one phoneme at a time from the input grapheme sequence.

2.6.3.3 Pronunciation lexica

Pronunciation lexica contain entries that map from word spellings to their pronunciations, usually encoded as a sequence of phonemes. Additional linguistic information is usually provided such as part of speech, morpheme boundaries, syllable boundaries, and syllable stress indicators; which help pronunciation disambiguation and provide contextual information for improving waveform generation. Figure 2.7 shows pronunciation lexica entries from CMUDict (Weide, 1998) and Combilex (Fitt and Richmond, 2006) and how both phonemic and other linguistic information can be encoded.

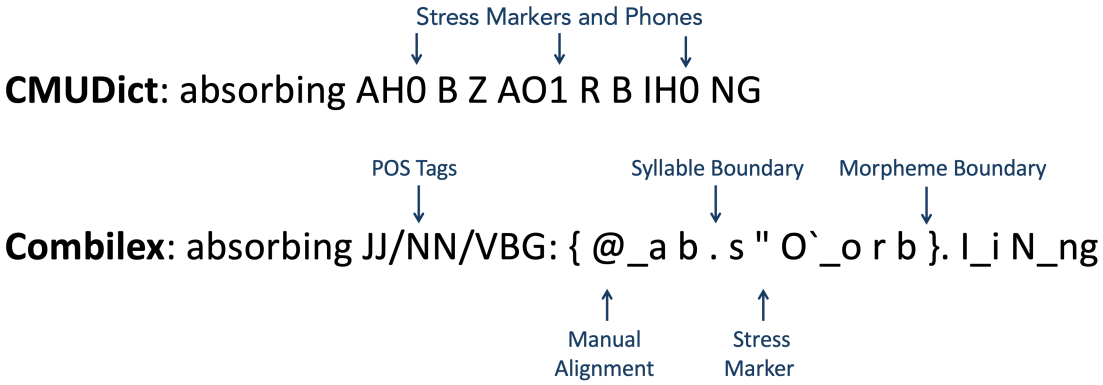

Figure 2.7: Example pronunciation lexica entries from CMUDict and Combilex, that map from a word's surface graphemic form, to a sequence of phonemes, with additional auxiliary markup such as syllable boundaries, morpheme boundaries, stress markers, and part-of-speech tags. Image credit to Jason Taylor's first year review document, no longer available online.

When the accuracy of pronunciation is crucial, entries of a lexicon are transcribed manually by linguists. Although this can be a very expensive process, the cost can be somewhat amortised: for example, many entries in Unilex (Fitt, 2018), an open source lexicon for TTS, were created systematically and thus cheaply from the pronunciations of lemmas (i.e. the base form of words without inflections). For example, given that we have the phoneme sequence for a regular word such as 'want' and the phoneme sequence for the morphemes '-s', '-ed', '-ing', we can easily generate the pronunciations for 'wants', 'wanted', and 'wanting' by combining their phoneme sequences together. This approach however requires comprehensive linguistic knowledge of which words can be conjugated or inflected in a regular manner.

Despite their cost, pronunciation lexica can be not completely correct or internally consistent. Pronunciations for words within a lexica often have murky provenance as they are typically added gradually over many years by many contributors, who may transcribe words differently from each other. Just as in the construction of datasets used for other machine learning tasks, 'inter-annotator agreement' is not guaranteed. Furthermore, errors can be introduced into a lexicon when entries are generated by a G2P model, which will most likely mispronounce unpredictably spelt words. Additionally due to the high inertia associated with switching the phoneme set of a large lexicon, it is extremely costly to make

fundamental improvements to how pronunciations are represented in a lexicon.

Production systems, which require high levels of correctness, need extremely large pronunciation lexica. Such lexica may contain as many as 435,000 entries (van Esch et al., 2016). Building a large lexicon is extremely expensive because linguists need to transcribe new entries and ensure the correctness and consistency of existing ones. Regular checking of existing entries in a lexicon is necessary as the way language is used is constantly evolving. Therefore, such lexica are only economically viable for the world's most widely used languages. This creates a challenge for less widely spoken languages, where financial resources and linguistic talent are often scarce, and high-quality TTS technology is not commercially viable. This reduces the accessibility of high-quality TTS.

2.6.4 End-to-end and grapheme-based text-to-speech

As discussed earlier in this section, most TTS applications require accurate pronunciations. The majority of TTS systems therefore use phonemes as an intermediate representation between the input text and the acoustic model. Predicting phonemes from the input text requires a pronunciation lexicon and a G2P model, both of which are costly and time-consuming to create. In contrast, using graphemes as input to the acoustic model is a newer TTS paradigm enabled by end-to-end TTS systems (Shen et al., 2018b; Tachibana et al., 2018a; Li et al., 2019; Łańcucki, 2021). This paradigm potentially eliminates the need for phoneme-based resources, and thus offers a promising solution for scaling TTS to many more of the world's languages. End-to-end systems consist solely of neural modules that can be trained jointly using only text-audio paired data to perform TTS. This is in stark contrast to traditional TTS systems that require heuristically engineered front-end and back-end components.

2.6.4.1 Architecture of end-to-end text-to-speech systems

When compared to traditional TTS paradigms that cannot achieve end-to-end TTS, the Seq2Seq neural network topology that first achieved this goal is surprisingly far simpler (Wang et al., 2017b). Seq2Seq TTS predicts waveform samples or mel-spectrogram frames from grapheme inputs using neural encoders and decoders. Figure 2.8 shows an overview of Tacotron 2, the first Seq2Seq TTS model to enable fully end-to-end training.

Neural encoders, which can be considered the 'front-end' of end-to-end TTS models, progressively form representations from input text that more closely resemble output speech. These encoders are trained from 'scratch'; that is they are trained in a purely data-driven way on text-audio pairs only. The neural 'front-ends' of such Seq2Seq models are not similar to the pipeline-based front-ends that generate human understandable contextual linguistic features. Instead Seq2Seq models employ deep Recurrent Neural Network (RNN) (Rumelhart et al., 1986), Convolutional Neural Network (CNN) (Krizhevsky et al., 2012), or Transformer-based (Vaswani et al., 2017) encoders that learn to extract higher level representations of the input text that provide information useful for predicting the output speech. The use of RNNs and/or the attention mechanism is crucial for grapheme-input TTS as these mechanisms can deal with the non-monotonic alignment between graphemes and speech sounds. Additionally text encoders, as investigated by Bailly et al. (2023) for example, can automatically learn allophonic variation from grapheme inputs only.

The decoder or 'acoustic model' then incrementally generates speech frames conditioned on both previously generated speech and relevant linguistic information output by the text encoder selected through the dynamic attention mechanism that can consider non-monotonic mappings between input text encoder representations and the output speech frame currently being predicted.

2.6.4.2 Pronunciation errors in end-to-end text-to-speech

Grapheme-input end-to-end TTS (hereby referred to as grapheme-input TTS) is a promising paradigm for future TTS systems. As grapheme-input TTS does not require language-specific resources, its key advantage is its ability to build a voice for any language without the need for an expensive front-end. However, one major downside is its tendency to make pronunciation errors since it is impossible to guarantee the correct pronunciation of unseen words with unfamiliar pronunciation. This limits its real world applicability. Such errors occur because natural languages typically do not exhibit one-to-one mappings between graphemes and speech sounds (Marjou, 2019; Pereira, 2019; Perquin et al., 2020), which, in a similar fashion to the task of G2P prediction, makes the implicit pronunciation prediction task inside end-to-end TTS models error-prone (Taylor and Richmond, 2019b; Yasuda et al., 2021b).

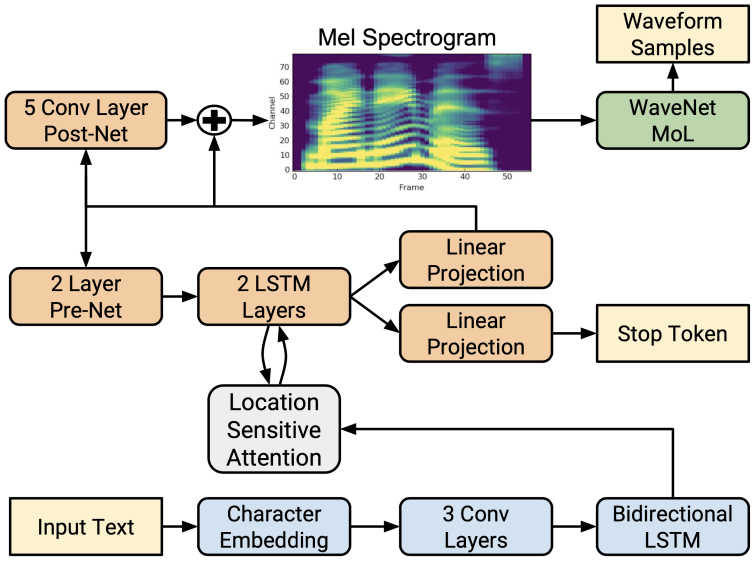

Figure 2.8: Overview of the sequence-to-sequence with attention architecture for Tacotron 2 that was the first TTS model to enable fully end-to-end training from raw text to waveform samples. This image is taken from the original paper by Wang et al. (2017b).

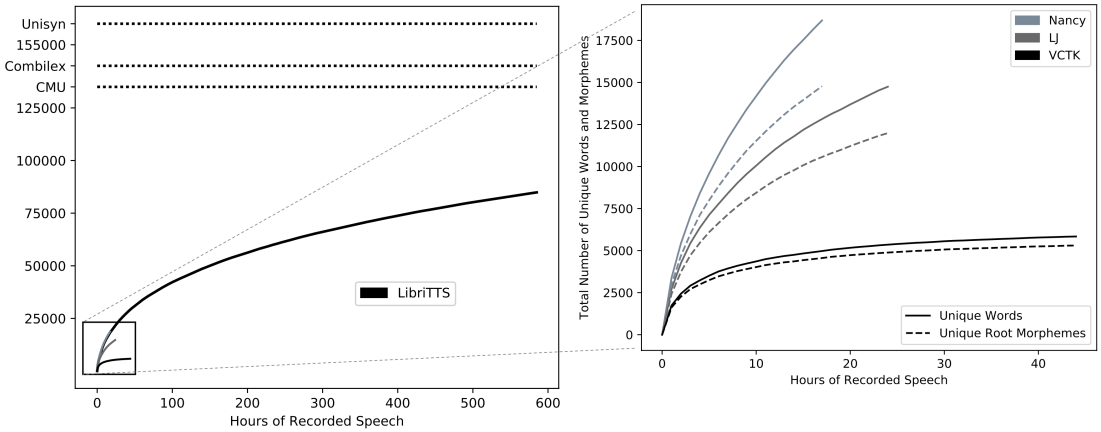

Figure 2.9: A plot of the cumulative coverage of unique word types in TTS corpora and pronunciation lexica. Figure taken from Taylor and Richmond (2019a).

Evidence from multiple studies suggests that Seq2Seq models learn a weaker joint pronunciation and acoustic model from graphemes than from phonemes, resulting in overall lower naturalness (Kastner et al., 2019; Shen et al., 2018a; Taylor and Richmond, 2019a). A potential reason for this finding is that the implicit G2P model learned by grapheme-input TTS is less accurate than explicit G2P models. Grapheme-input TTS models implicitly learn pronunciation patterns from TTS

training corpora that contain far fewer word types than a typical pronunciation lexicon (e.g. 14,750 word types in LJ Speech vs. 165,000 in Unisyn (Taylor and Richmond, 2019a)). This means that when compared to an explicit G2P model trained on a whole lexicon the implicit G2P model of a grapheme-input TTS system will be exposed to fewer irregular words and foreign loanwords and thus fewer irregular G2A relations found in words such as 'pneumatic', 'Flaubert' or 'baguette'. Figure 2.9 plots the total number of word types in commonly used TTS corpora and pronunciation lexica.

Subsequently, in order to realise the full potential of grapheme-input TTS, its tendency to make pronunciation errors must first be solved. One possible solution is to increase the number and variety of word pronunciations encountered during training. However, due to the Zipfian distribution of words in natural language, typical TTS corpora have limited word type coverage in comparison to pronunciation dictionaries Although one could theoretically improve grapheme-input TTS pronunciations by training on vast amounts of text-audio data, this would complicate model deployment and inflate data collection costs to an insurmountable level. Additionally, and perhaps more crucially, this data-centric approach would not guarantee any practical improvement as it does not solve the unpredictability intrinsic to the mapping between graphemes and speech sounds. As such, simply training with more data is not a viable solution. Furthermore, while it is theoretically possible to obtain accurate pronunciations by training grapheme-based TTS on a dataset that covers *all* possible word types, this approach is impractical, and as new words are constantly being created, it would be very difficult to keep such a dataset up-to-date and costly to constantly retrain models.

2.6.4.3 Controlling and correcting pronunciations: grapheme-based text-to-speech

Due to the aforementioned lack of pronunciation control when using grapheme inputs, neural TTS systems, both in research and production, often use rich contextual features as generated by a front-end. Although the use of a front-end enables a degree of pronunciation control, it does not guarantee it.

Explicit G2P will inevitably make mistakes because the vast majority of orthographies are not phonemic, thus it is not enough to simply 'solve' G2P in order to achieve robust pronunciation control. This means that a dictionary-based

pronunciation lookup approach must be adopted in order to correctly pronounce irregular words. In this thesis, we explore novel dictionary-based approaches to pronunciation control for grapheme-input TTS that *do not* involve linguistic front-ends that depend on phoneme resources.

2.7 Vocoders for text-to-speech

Vocoders are an important component of TTS systems. They generate speech waveforms from some representation of speech predicted by the acoustic model, whether that be hand engineered or learned.

A common traditional vocoder, which we employ in this thesis, is the Griffin-Lim algorithm (Griffin and Lim, 1984) that iteratively "inverts" a magnitude spectrogram to generate a waveform. However, the algorithm has two main drawbacks. The first is that many iterations are needed to generate high quality audio. The second is that its speech quality is severely lacking and is not comfortable to listen to.

Another signal processing based vocoder is STRAIGHT introduced by Kawahara et al. (1999). The STRAIGHT vocoder operates by first analysing the input speech signal to extract fundamental frequency (pitch), spectral envelope, and aperiodic components. These parameters can then be independently adjusted to achieve various transformations, such as pitch shifting, timbre modification, and time-scale changes. Finally, the modified parameters are used to reconstruct the speech signal, ensuring that it remains natural and clear despite the alterations. This process allows STRAIGHT to decompose, modify, and reassemble speech while maintaining high quality and intelligibility.

In recent years, neural vocoders have gained prominence since their output is very high quality, often indistinguishable from ground-truth speech recordings. Furthermore, recent architectural improvements have made neural vocoders competitive in inference speed, often being faster than real time. Two vocoders used in this thesis for their fast inference and high quality output are WaveRNN (Kalchbrenner et al., 2018) and HifiGAN (Kong et al., 2020).

2.8 Speech-based self-supervised learning

Self-supervised learning (SSL) is a recent paradigm in speech technology that focuses on building models that can discover the underlying structure of speech without requiring labels. These models are trained in a *self-supervised* fashion, often predicting masked sequences of speech, contrasting positive samples from negative samples, or reconstructing speech, which encourages them to contextually represent their input at a higher level.

One of the primary benefits of self-supervised learning is that it can reduce the amount of paired or labelled data needed to train a model to perform a downstream task. For example, a SSL model can be pretrained solely using speech data in a self-supervised fashion, and then finetuned using a smaller amount of labelled data to perform tasks such as ASR (Baevski et al., 2020; Hsu et al., 2021), or emotion recognition (Yang et al., 2021).

In this thesis, rather than finetuning speech-based SSL models for the task of TTS, we utilise their ability to generate phone-like representations. SSL models tend to produce speech representations that correlate with a variety of linguistic and speech features. These representations can then be converted to phone-like units via a discretisation method such as k-means clustering. Prior work has shown that discrete SSL speech representations contain phone or subphone information (Wells et al., 2022), while discarding non-segmental information such as channel condition and speaker identity, making them effective for applications such as voice conversion (Polyak et al., 2021), spoken language modelling (Lakhotia et al., 2021), or speech-to-speech translation (Lee et al., 2021a). In a similar vein, we take advantage of these properties of discrete SSL speech representations, using them to represent word pronunciations in a largely speaker-agnostic fashion in Chapters 6, 7, and 8.

2.8.1 Generating discrete self-supervised learning speech representations

There are two main methods for generating discrete SSL speech representations. The first method is directly predicting discrete latent features within the model, and the second method is to discretise continuous features output by some layer of the model.

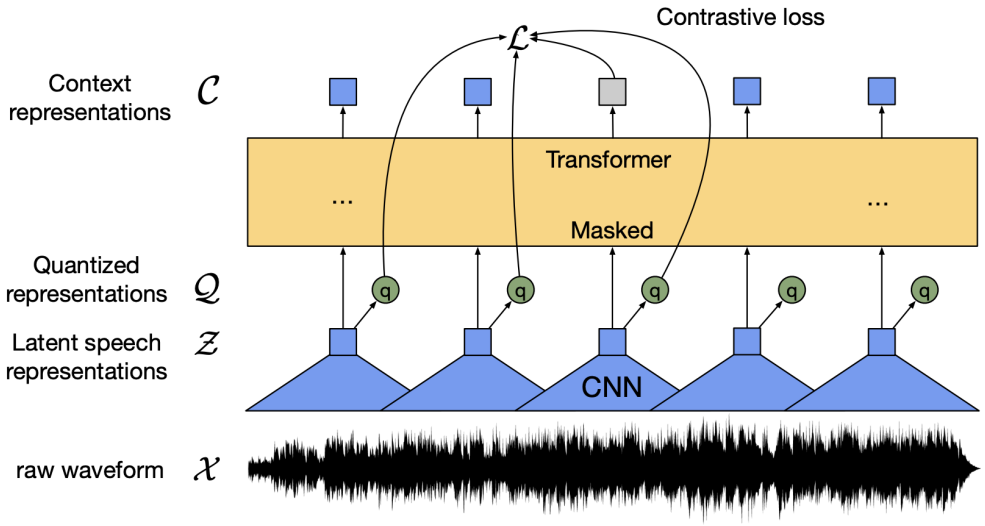

Figure 2.10: An overview of the wav2vec 2.0 architecture. Image credit belongs to the original paper by Baevski et al. (2020).

The first method is exemplified by the wav2vec 2.0 architecture (Baevski et al., 2020). During pretraining, it encodes speech using convolutional layers to produce latent representations, which are then discretised into speech tokens using Gumbel softmax (Jang et al., 2016). The model is optimised using a contrastive loss function, which encourages the continuous output of a context network to be close to masked discrete speech tokens in terms of cosine similarity. This loss function encourages the context network to contextually represent the input speech at a more abstract level. An overview of the wav2vec 2.0 architecture can be found in Figure 2.10.

The second method, as seen in HuBERT (Hsu et al., 2021), involves discretising continuous speech features into discrete tokens using a clustering model (such as k-means clustering) *after* pretraining. The model is pretrained so that its outputs are predictive of the correct clusters for a masked segment of the input, based on the surrounding context. The optimisation of HuBERT involves a cross-entropy loss function between the cluster labels and a predictive layer's cluster predictions. The training of HuBERT is iterative, using cluster labels from the previous iteration of the model to train the speech encoder of the current iteration using a masked cross-entropy loss function. In the first iteration, since the model is uninitialised, k-means clustering over MFCC features is used to create the first round of cluster labels. In practice the iterative training procedure of HuBERT is more stable than that of wav2vec 2.0, making it easier to reproduce.

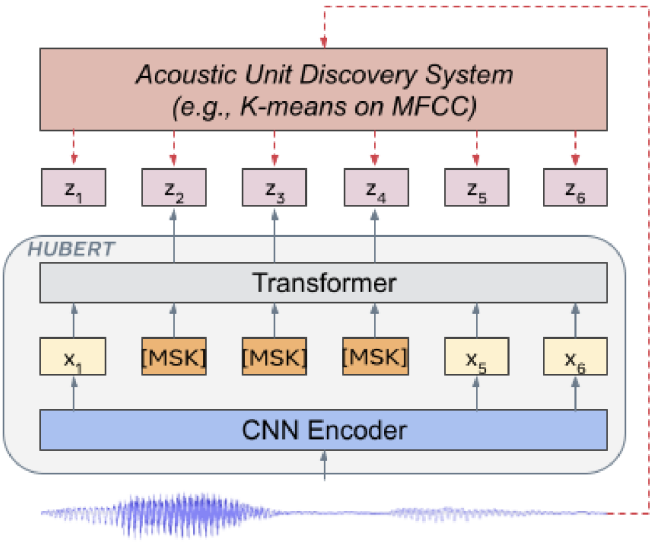

Figure 2.11: An overview of the HuBERT architecture. Image credit belongs to the original paper by Hsu et al. (2021).

Once HuBERT is trained, features from any layer can be clustered, and then discretised to perform any given task. For example, one can extract phone-like units/embeddings by clustering representations from the HuBERT layer whose continuous representations are most predictive of phoneme classes. The centroids of each cluster, or the cluster IDs, can then be used as phone-like representations for a downstream task. An overview of the HuBERT architecture can be found in Figure 2.11.

2.8.2 Information content of discrete self-supervised learning speech representations

In this subsection, we explore prior research on the informational content of discrete SSL speech representations, specifically focusing on discrete HuBERT representations, as these are adopted for our experiments in this thesis.

Phonetic content: Previous work has demonstrated that discrete speech representations, when the number of k-means clusters is relatively small (i.e. 50 to 100 clusters), mostly contain information related to phone or subphone units. The original HuBERT paper, and several others, have performed correlation analyses between the k-means clusters and the actual phone units (as determined by an ASR system), and found that there is a high correlation between them, and that this correlation improves at later layers of the model (Hsu et al., 2021; Sicherman

and Adi, 2023). Wells et al. (2022) further explored discrete SSL speech units, breaking down their analysis into broad phonetic classes (e.g. plosives, fricatives, etc.). They find that discrete units can correspond to sub-phonetic events, for example, one discrete unit is found to represent plosive closures across *multiple* phonemes.

Prosodic and speaker content: While no prior work to the best of our knowledge has directly examined the prosodic or speaker content of discrete representations, results in TTS and voice conversion suggest that discrete speech units do not contain much prosody information such as f_0 or energy or speaker information. Voice conversion (VC) (Polyak et al., 2021) and TTS (Huang et al., 2023) systems have used discrete speech units as an intermediate feature, either as speaker-agnostic features for VC, or for reducing the amount of paired text-audio data required for TTS. Such systems have shown that expressivity can be improved if one extracts f_0 and energy algorithmically during training as auxiliary features for conditioning the acoustic decoder or vocoder. At inference time, these f_0 and energy features are extracted from the source speaker when performing voice conversion, and are predicted from the input text for TTS. Speaker identity embeddings or encodings are also a commonly used additional auxiliary feature.

2.9 Data

In this section we discuss the datasets used in our experiments, LJ Speech and VCTK (Voice Cloning Toolkit).

The LJ Speech dataset consists of 13,100 short audio clips of a single female speaker reading passages from various non-fiction books, totalling approximately 24 hours of audio (Ito and Johnson, 2017). It includes corresponding text transcriptions, making it suitable for tasks such as TTS and speech recognition. The VCTK (Voice Cloning Toolkit) dataset contains speech data from 109 native speakers of English with various accents, each reading approximately 400 sentences, resulting in about 44 hours of speech (Yamagishi et al., 2019). Both datasets provide high-quality, professionally recorded audio with precise alignment between audio clips and transcriptions, ensuring clear and accurate pronunciations. The VCTK dataset offers a wide range of accents and pronunciations, making it valuable for evaluating models across different speaking styles.

The English-based LJ Speech and VCTK datasets have been shown to offer cross-linguistic applicability, as techniques developed and tested on them can often be adapted to other languages, allowing for evaluation in different phonetic and phonological contexts. Models trained on these datasets can be fine-tuned for other languages, leveraging robust pronunciation and acoustic models initially developed with English data, thus expediting the creation of accurate pronunciation models for various languages. However, techniques effective for English may require modifications to handle tonal languages like Mandarin or languages with a rich set of phonemes like Hindi. Additionally, results based on English datasets might not fully capture the pronunciation challenges in languages with different scripts, phoneme inventories, or prosodic features. Nevertheless, we utilise these datasets due to their ease of use, availability, and suitability as testing grounds to demonstrate our methods.

2.10 Conclusion

In this chapter, I have provided a thorough background of the scientific and methodological foundations of this thesis. The rest of this thesis, starting from chapter 3 and ending in 8, details the research projects that were undertaken in order to answer my core research question, *"Can precise control over the pronunciations of a TTS system be achieved when using few or no linguistic resources such as phoneme transcriptions?"*.

Chapter 3

Robustness of grapheme-input text-to-speech trained on imperfect transcriptions

This chapter represents an extended version of my research presented in *Investigating the robustness of sequence-to-sequence text-to-speech models to imperfectly-transcribed training data*, which was published in the proceedings of Interspeech 2019 (Fong et al., 2019b). This was joint work done in collaboration with Pilar Oplustil Gallegos, Zack Hodari, and Simon King.

3.1 Introduction

In the broad landscape of Text-to-Speech (TTS) research, one of the more under-explored areas is the control of pronunciation in grapheme-input end-to-end systems. This area is particularly relevant for low-resource languages, where traditional phoneme-based pronunciation resources are often unavailable or incomplete. Specifying pronunciation using grapheme inputs offers a viable avenue for the development of TTS systems that can better serve a more diverse range of language communities. Against this backdrop, the focus of this chapter takes on additional significance.

As a preliminary step in the journey of this thesis, we first investigate the feasibility of training end-to-end systems using *imperfect data*–a scenario frequently

encountered in resource-constrained conditions. We train Sequence-to-Sequence (Seq2Seq) TTS models on data with simulated transcription errors, and evaluate them for overall quality and number of mispronunciations.

This empirical examination of training data quality holds a pivotal role in investigating the effectiveness of Seq2Seq TTS systems. While these models exhibit promise in replicating natural speech when provided with ample and accurately transcribed training data (Sotelo et al., 2017; Wang et al., 2017b; Ping et al., 2018), their performance can significantly degrade as the available training data diminishes, not only in terms of naturalness but also in the occurrence of synthesis errors (Chung et al., 2019; Latorre et al., 2019a). To address this challenge, researchers have employed various strategies aimed at mitigating the constraints of limited data. These strategies include pooling data from multiple speakers and incorporating speaker embeddings (Ping et al., 2018; Jia et al., 2018; Latorre et al., 2019a). Additionally, some have explored the pre-training of TTS models using unpaired text or audio data, enhancing generalisation capabilities and mitigating the risk of overfitting to smaller, higher-quality datasets (Chung et al., 2019).

Yet, at the time our work was done, an expansive, albeit less explored, source of data is 'found data', which can be obtained from audiobooks, podcasts, and other multimedia platforms. While abundant, this data frequently contains transcription errors, arising from Automatic Speech Recognition (ASR) systems and human inaccuracies such as manual transcription errors or speaker inconsistencies. Traditional TTS systems, i.e. those not using a neural Sequence-to-Sequence architecture, tackle transcription inaccuracies through various techniques, a topic further discussed in Section 3.2. However, the resiliency of neural Seq2Seq TTS models against transcription imperfections remains largely unexplored. The role of the attention mechanism within Seq2Seq models presents a double-edged sword. On one hand, it can dynamically disregard errors in transcripts during the training phase, in contrast to the standard forced alignment method, which can only align incrementally. On the other hand, this very flexibility introduces the risk of the model skipping words during the synthesis stage.

Consequently, the primary objective of this chapter is to empirically evaluate the sensitivity of Seq2Seq TTS models to the quality of the transcript of the training data. This study constitutes an essential cornerstone of our goal of constructing end-to-end TTS systems capable of reliable pronunciation control. Through our

experiments, we aim to determine whether Seq2Seq TTS models can indeed be robustly trained on both clean and imperfect transcripts. Our findings contribute towards a more sustainable and cost-effective approach to extend the reach of high-quality TTS technologies across linguistically-diverse communities.

Note, as this research was carried out before the introduction of duration based neural TTS models such as FastSpeech we do not explore the impact of transcription errors on these types of models.

3.2 Related work - traditional data robust text-to-speech paradigms

Extensive research has been conducted on the challenges and solutions associated with building various kinds of speech synthesis systems using imperfect data. This earlier traditional paradigms like unit-selection and Hidden Markov Model (HMM)-based Statistical Parametric Speech Synthesis (SPSS), as well as more modern neural paradigms.

In the field of TTS, the primary emphasis has centred on employing universal data cleaning techniques that are applicable across various TTS model types. This process often involves removing potentially problematic utterances from the training data to enhance TTS performance. For instance, in work by Stan et al. (2013) and Braunschweiler et al. (2010), lightly supervised alignment models were employed to identify training utterances that possess entirely accurate transcriptions, particularly when working with audiobook data. Training on only accurately transcribed utterances enhances TTS quality.

Aside from these universal data cleaning techniques, there are also paradigm-specific error-handling mechanisms that to further improve TTS performance. In **unit selection**, forced alignment associates phonetic labels with corresponding segments of audio, thereby detecting transcription mismatches (Clark et al., 2007a). At synthesis, the system relies on the join cost, and optionally an acoustically-based target cost (in hybrid unit selection) to minimise the likelihood of choosing inappropriate units. **HMM synthesis** systems use beam pruning during Baum-Welch estimation to reduce the influence of mistranscribed data. HMM-based SPSS systems also exhibit robustness when trained on phonetically-imbalanced

corpora which are recorded under less-than-ideal conditions, often outperforming unit selection models (Yamagishi et al., 2008).

In this work, we utilise a newer attention-based TTS model. The attention mechanism, originally developed for machine translation Bahdanau et al. (2015), dynamically aligns source language and target language text during training, negating the need for an external alignment model. This functionality has been highly beneficial to machine translation as pairs of languages often exhibit different syntactical orderings. This ability to align two non-monotonically aligned sequences is promising for robust TTS. The mechanism potentially has the ability to skip over irrelevant elements in the input text and/or the output speech making TTS robust to specific error scenarios. This departure from traditional TTS methods presents a promising avenue for addressing data imperfections without relying on extensive error-handling strategies.

3.3 Experiments

The primary objective of our experiments is to evaluate the robustness of a sequence-to-sequence TTS system in the presence of transcription errors. To achieve this, we designed a set of experimental conditions that subject the model to both manual and automatic annotation errors. The results will offer insights into how the model reacts to inaccuracies typically encountered in real-world transcribed text. We aim to answer the following research questions:

- How does the pronunciation accuracy of the system change when trained using transcripts containing various types of word-level errors: additions, deletions, and substitutions?

- How does the pronunciation accuracy of the system change when trained using transcripts containing errors made by an automatic speech recognition (ASR) system?

In the following sections, we will describe, our hypotheses, the dataset used, the process of simulating transcription errors, and the architecture of our Seq2Seq TTS model.

3.3.1 Hypotheses

Table 3.1 outlines our hypotheses regarding the use of training data with transcripts subjected to the corruption methods described in Section 3.3.3.

3.3.2 Data

In all experiments in this chapter, we exclusively employ the LJ Speech dataset[1]. This dataset consists of approximately ∼24 hours of audio comprising 13,100 sentences, all of which are narrated by a single American female speaker while reading 7 non-fiction books. The provided transcriptions were already normalised for numbers, ordinals, acronyms, and monetary units. The training text retains punctuation such as commas and periods. As for additional processing, we perform only minimal adjustments to the training text, converting uppercase characters to lowercase and adding tokens to mark word boundaries, punctuation, and the start and end of sentences. On average, each sentence in the dataset contained ∼17 words (∼100 characters).

3.3.3 Simulating transcription errors

To investigate how Seq2Seq TTS models deal with transcription errors we required an appropriate source of training data. However, the scarcity of naturally occurring data with transcription errors posed a challenge. To address this limitation, we deliberately introduce errors through artificial means. This simulated method allows us to establish a controlled and replicable testing environment for the assessment of Seq2Seq TTS models' robustness.

3.3.3.1 Simulating errors in manual transcription

To replicate word level errors that can occur during manual transcription, we implemented three methodologies:

- **Add 5 words**: Five random words from other utterances in the training data, with lengths within one character of the average word length (7 characters), are randomly inserted into the sentence in random locations. For example, "the examination and testimony of the experts" could be corrupted to "**their**

[1] https://keithito.com/LJ-Speech-Dataset/

Table 3.1: Hypotheses

Clean vs Corrupted Data	
H_1	Systems trained on clean data outperform those trained on corrupted data.

Amount of Data	
H_2	Systems trained with more data perform better than those trained with less data.

Relative Performance Between Corruption Methods	
H_3	Systems trained on data corrupted by **Add 5 words** will exhibit the highest performance because the attention mechanism will learn to skip added words.
H_4	Systems trained on data corrupted by **Delete 5 words** will perform worse than those with **Add 5 words** because the attention mechanism will not find relevant information in the input sequence to explain certain sequences of acoustic frames.
H_5	Systems trained on data corrupted by **Replace 5 words** will show the poorest performance because the attention mechanism will be forced to use incorrect input to explain certain acoustic frames.
H_6	Systems trained on ASR transcription output will outperform those with **Replace 5 words** and **Delete 5 words** because the attention mechanism will find plausible (though not entirely correct) input to explain all acoustic frames.

because the **automobile** examination and testimony of **only** the experts **peace**".

- **Delete 5 words**: Five words are randomly selected and removed from each sentence, provided that at least one word remains. For example, "Many animals of even complex structure which live parasitically within others are wholly devoid of an alimentary cavity", could be corrupted to "Many _ of even _ structure which live parasitically _ others _ wholly devoid of an _ cavity".
- **Replace 5 words**: Five words in the sentence are substituted with words of equal length, each randomly drawn from a different training utterance. For example, "Many animals of even complex structure which live parasitically within others are wholly devoid of an alimentary cavity", could be corrupted to "Many **as** of even **placebo** structure which live parasitically **ghost** others **yellow** wholly devoid of an **placate** cavity".

We chose to corrupt with 5 words each time in order to present a sufficient challenge to the Seq2Seq TTS model during training. We also considered introducing other forms of manual transcription errors such as typos but lacked a real world dataset with which to generate realistic errors from the TTS training data.

3.3.3.2 Simulating errors in automatic transcription

To obtain automatic transcription errors, we processed each audio utterance using a Hidden Markov Model Deep Neural Network (HMM–DNN) ASR system implemented in Kaldi Povey et al. (2011). From the generated lattice, we selected the top 50 hypotheses and calculated the Word Error Rate (WER) for each. We then chose the hypothesis with the highest WER for each utterance. Unlike the manually induced errors described in 3.3.3.1, errors in automated transcription are acoustically plausible. For instance, the phrase "I need to buy some new shoes" could be erroneously transcribed as as either "I need to pie some nude chews" or "Eye knead two by sum new sous" by the ASR system.

3.3.4 Model architecture

Our model is a variant of the deep-convolutional TTS system (DC–TTS), introduced by Tachibana et al. (2018b), a convolutional encoder-decoder with an autoregressive structure in the decoder[2]. One advantage of using DC–TTS for

[2]Our implementation is based on https://github.com/Kyubyong/dc_tts

our experiments lies in its training speed. This is attributable to the use of convolutional layers, as opposed to recurrent layers, in both the encoder and decoder. These are better suited for parallel computation using GPUs, resulting in a 19.2-fold increase in speed compared to Tacotron 2. Despite this, it produces synthesised speech of comparable quality.

To streamline training, we adhere to the standard DC–TTS procedures by training two separate models independently using ground truth data. The Seq2Seq component of the model, known as Text-to-mel (T2M), predicts 80-band mel filter bank features (MFB) from graphemes. We extract this intermediate representation from the magnitude spectrogram (captured at 80 frames per second) using pre-emphasis, mel-scale bins, and a timescale reduction by a factor of 4, resulting in a frame rate of 20 Hz. We refer to these lower framerate mel filter bank features as coarse MFBs. Following the T2M stage, the Spectrogram Super-Resolution Network (SSRN) employs transposed convolutions—most commonly used to increase the spatial resolution of inputs—to reconstruct the full-resolution magnitude spectrogram. Placed together in sequence, these two models perform grapheme to magnitude spectrogram prediction.

DC–TTS incorporates four trainable modules, depicted in blue in Figure 3.1. The T2M model itself consists of three modules that together predict the coarse MFBs, denoted as $\hat{Y}$, and are trained independently from SSRN. Here is a brief overview of key parts of DC–TTS:

- TextEnc: This module encodes N one-hot characters into a sequence of d dimensional keys K and values V for use in attention querying. Typically, attention learns a projection $V \to K$.
- AudioEnc: This module encodes T coarse MFB frames into d dimensional queries $Q_{1:T}$.
- Attention: Utilising multiplicative attention (Luong et al., 2015a), this module determines the alignment $A_{1:t} = \text{softmax}(K^\top Q_{1:t} * \sqrt{d})$, which is used to calculate the result $R_{1:t} = A_{1:t} V$. During synthesis, attention is forced to be monotonic and can only skip forward a maximum of 3 encoded characters per decoder time-step.
- AudioDec: This module predicts the current coarse MFB frame $\hat{Y}_t$ based on $R_{1:t}$ and $Q_{1:t}$. Note, the decoder works in an autoregressive manner and encodes

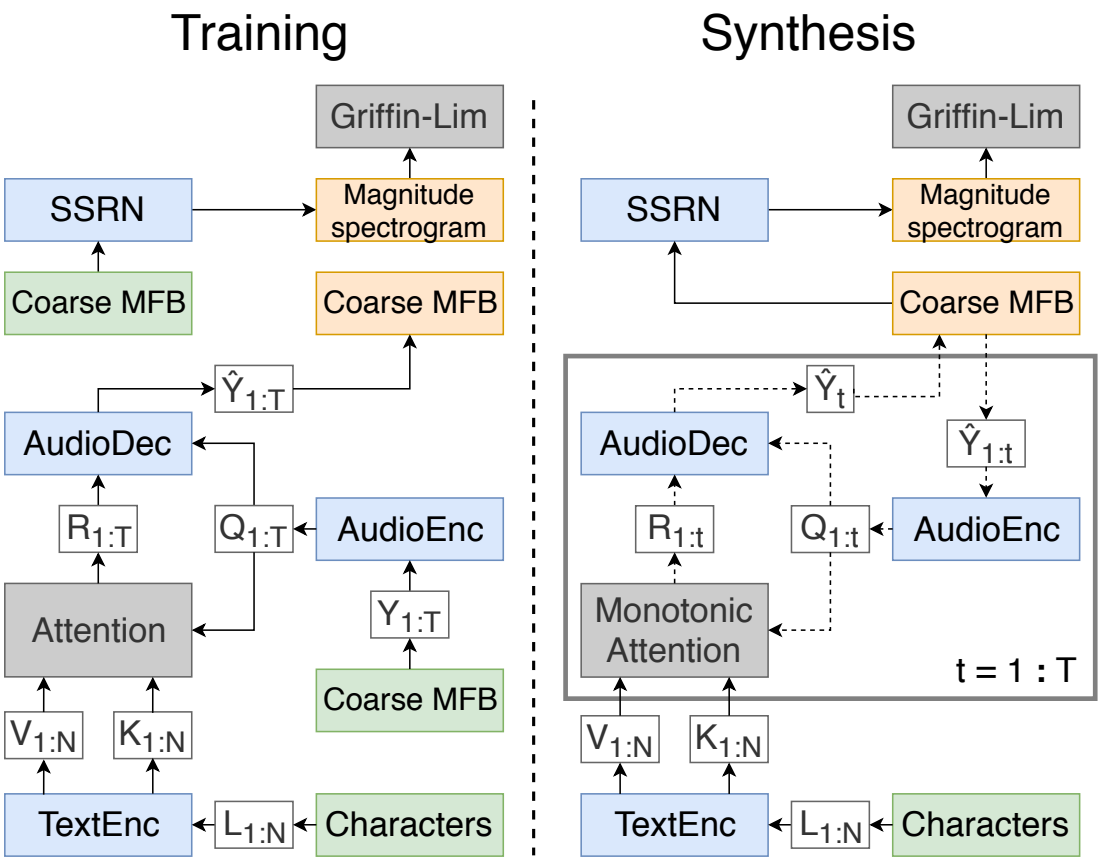

Figure 3.1: Architecture of the Deep-Convolutional Text-to-Speech system (DC–TTS). Components are color-coded as follows: Blue modules are trainable and consist of convolutional layers; Grey elements represent non-trainable computational operations; Green elements signify the input features; Orange elements represent the predictions made by the model. Plate notation over $t = 1 : T$ indicates the autoregressive nature of the decoder, looping from the first to the T^{th} timestep during synthesis.

only up to frame t, rather than all T frames.

- SSRN: The Spectrogram Super-Resolution Network upsamples the coarse MFB in time by a factor of 4 using transposed convolutions and reconstructs fine-grained frequency domain information.
- Griffin-Lim: This component finds a plausible phase spectrogram (Griffin and Lim, 1984) for the magnitude spectrogram predicted by the SSRN. After post-emphasis, an inverse FFT reconstructs the waveform.

In DC–TTS, the modelling of context on the text input side takes two primary forms: it is either captured within the convolution's receptive field or summarised through the attention mechanism with $R_{1:t} = A_{1:t}V_{1:N}$.

Within the autoregressive section of DC–TTS's T2M component, specifically AudioEnc and AudioDec, *causal* convolutions are employed. This choice is driven by the fact that during synthesis, we cannot utilise *future* acoustic context. On the other hand, TextEnc and SSRN, which have access to future context (characters or coarse MFBs), utilise *non-causal* convolutions.

Our model architecture bears a strong resemblance to the one described in Tachibana et al. (2018b). However, there are noteworthy differences: we forego the use of guided attention, as our objective is to equip the model with the capability to handle transcription errors, which would disrupt the roughly monotonic alignment between input graphemes and output acoustic frames. Additionally, we incorporate layer normalisation immediately after all non-transposed convolutions, and our dropout rate is set at 0.05.

During training, we initialise the learning rate to 0.001, progressively increasing it for the first 4000 batches before subsequently decaying it proportionally to the inverse square of the batch number, following the approach outlined in Vaswani et al. (2017). Our batch size is 16, and we employ the Adam optimiser (Kingma and Ba, 2015). The loss function for both models comprises an L_1 loss and binary divergence, with equal weighting, relative to their respective predictions—coarse MFB ($\hat{Y}$) for T2M and magnitude spectrograms ($\hat{S}$) for SSRN.

$$\text{Loss} = \frac{1}{2}\left(L_1(\hat{Y}_{\text{T2M}}, Y) + L_1(\hat{S}_{\text{SSRN}}, S)\right) \tag{3.1}$$

3.4 Evaluation

3.4.1 Systems built for evaluation

To test these hypotheses, we constructed six Seq2Seq-TTS systems along with a reference voice. We employed each of the data corruption methods to generate four versions of corrupted transcripts. For each of these variants, we trained a distinct DC-TTS system. Specifically, 50% of the sentences were corrupted, affecting all even-numbered ones. Additionally, we trained a fifth system using the complete, uncorrupted dataset, and a sixth system was trained using only half of the sentences (all odd-numbered ones). This ensured that all training sets contained the same uncorrupted half of the full dataset, with the remaining half

being either uncorrupted, corrupted, or omitted. Chapter 50 of LJ Speech was left out of the training data as a test set. To synthesise the test sentences, all models utilised the same SSRN model, which had been trained for 500 epochs on all of the training data. It is worth noting that this model is independent of transcriptions and is not affected by any corruptions, hence we name our TTS systems 'T2M'. Here are the system names and the data used for training each of them:

$\text{T2M}_{\text{clean–100}}$: The full uncorrupted LJ Speech training dataset.

$\text{T2M}_{\text{clean–50}}$: Half of the training dataset.

T2M_{ADD}: Half uncorrupted utterances and half corrupted by the **Add 5 words** method.

T2M_{DEL}: Half uncorrupted utterances and half corrupted by the **Delete 5 words** method.

T2M_{REPL}: Half uncorrupted utterances and half corrupted by the **Replace 5 words** method.

T2M_{ASR}: Half uncorrupted utterances and half corrupted by ASR, which has an average Word Error Rate (WER) of 34.8%.

REF: Copy synthesis obtained by running ground truth coarse mel filter bank features through the SSRN model and then reconstructing waveforms with Griffin-Lim.

All systems underwent training for 250 epochs, except for $\text{T2M}_{\text{clean–50}}$, which was trained for 500 epochs. This approach ensured that the number of model updates during training was the same across all systems. The systems were trained using letters as input to the DC–TTS model.

3.4.2 Evaluation methodology: MUSHRA-like test

To evaluate each model's performance, we used a subset of 40 sentences from the 278 sentences contained in chapter 50 of the LJ Speech dataset. Several generated audios can be accessed on the samples page[3]. These sentences were not part of the training set. From the remaining sentences, we randomly selected 40 for our listening test. These sentences fell within a range of [-4 to +4] words from the mean sentence length, which, across the entire corpus, is 17 words.

[3]Speech samples are available here https://jonojace.github.io/IS19-robustness

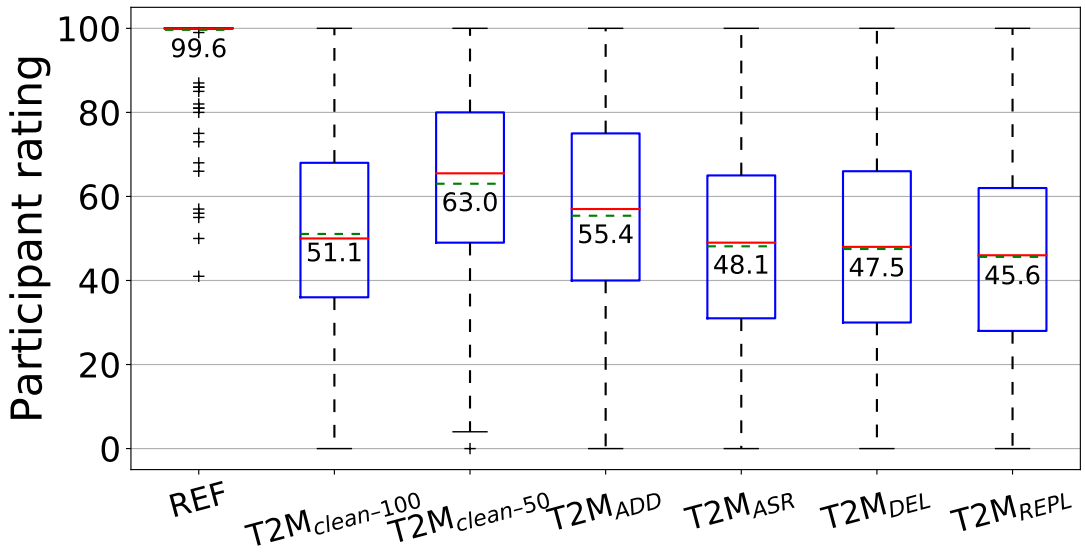

Figure 3.2: MUSHRA results. Solid red lines within each box represent the medians. Dashed green lines within each box denote the means (numerical values also provided in the figures). Top and bottom edges of each blue box illustrate the 25th and 75th percentiles. Whiskers indicate the range of the ratings, excluding outliers, which are marked with a +.

We constructed a listening test using BeaqleJS[4], following a framework similar to that outlined in ITU (2002), albeit without a lower-bound anchor, making it a MUSHRA-like rather than a standard MUSHRA test. Listeners received instructions to start by listening to the reference audio for each screen. They were told that the reference was hidden among the test sentences. Their task was to rate the 'quality' of each stimulus in relation to the reference. The participants were prompted with the following "Please rate each of the following stimuli on a scale of 0-100, where 100 represents the stimuli with the best quality in terms of naturalness and intelligibility" and were also provided with the correct text transcription for each sentence. To ensure thorough evaluation, participants could only proceed to the next screen once they had listened to all stimuli and rated at least one of them at 100. The order of systems on a screen and the order of screens for each participant were randomised. A total of 35 native English speakers, who were compensated for their participation, completed the test.

[4] https://github.com/HSU-ANT/beaqlejs

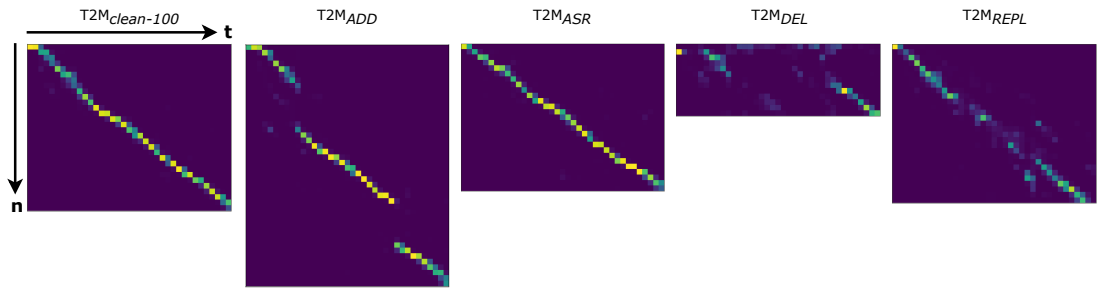

Figure 3.3: Attention matrices corresponding to the train dataset sentence with the original transcription, "In being comparatively modern." The encoder sequence (represented in characters) is displayed vertically, while the decoder output time (at a rate of 20 Hz) runs horizontally. These matrices maintain a consistent number of columns as they pertain to the same audio segment, but their number of rows varies due to transcription corruption. Each column illustrates the attention distribution over the input characters at a specific decoder step. Notably, during training, T2M_{DEL} and T2M_{REPL} occasionally encounter output segments that lack meaningful input, leading them to occasionally attend to the padding symbol (a zero vector). This behaviour can result in "babbling," where speech generation occurs without proper conditioning on text.

3.5 Results

The MUSHRA results are presented in Figure 3.2, where the hidden reference (REF) consistently received a perfect score of 100 from listeners, most of the time.

To assess the statistical significance of differences between pairs of systems, we conducted Student's t-tests for all 21 system pairs, applying Holm-Bonferroni correction (Holm, 1979) to account for the large number of comparisons. The results indicate significant differences (with $p < 0.0005$) for most of the pairs, except for the following exceptions:

T2M_{REPL} vs. T2M_{ASR} and T2M_{REPL} vs. T2M_{DEL}, which show differences at $p < 0.05$. T2M_{ASR} and T2M_{DEL} exhibit no significant difference. One particularly surprising finding is that T2M_{ADD} significantly outperformed $\text{T2M}_{\text{clean–100}}$, refuting H_1 (see Section 3.6.1 for further analysis).

Additionally, $\text{T2M}_{\text{clean–50}}$ was found to significantly outperform $\text{T2M}_{\text{clean–100}}$, which challenges H_2. However, we suspect that this outcome may have resulted from an unintended consequence of adjusting the training regimen. Since $\text{T2M}_{\text{clean–50}}$ had

half the data, we trained it for twice the number of epochs to maintain a similar number of weight updates.

Among the models trained on corrupted transcriptions, T2M_{ADD} consistently generated the highest-quality speech, followed by T2M_{DEL} and then T2M_{REPL}, confirming H_3, H_4, and H_5.

In contrast, the differences in quality between T2M_{ASR}, T2M_{DEL}, and T2M_{REPL} were marginal. This suggests that acoustically-plausible mistranscriptions are equally detrimental as arbitrary ones, refuting H_6.

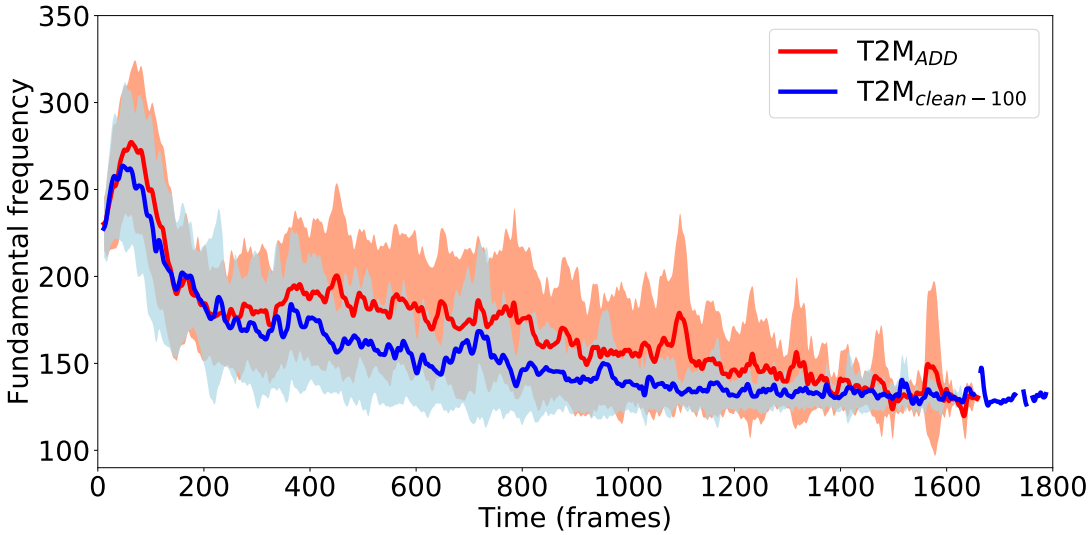

Figure 3.4: The figure displays per-frame F0 (fundamental frequency) statistics, including the mean and standard deviation, for the test sentences. The mean values are represented by lines, while the shaded regions denote the standard deviation.

3.6 Analysis

3.6.1 Effect of sentence length

Hypothesis H_1 was refuted, warranting further investigation. Extended listening revealed that the fundamental frequency (F0) behaviour in $\text{T2M}_{\text{clean–100}}$ was unnatural. In this case, F0 declined too rapidly and failed to reset, resulting in a very low value maintained until the end of the sentence. Figure 3.4 illustrates how $\text{T2M}_{\text{clean–100}}$'s F0 flattens out more quickly and exhibits a narrower standard deviation compared to T2M_{ADD}. This discrepancy is a potential explanation for the lower quality score observed for $\text{T2M}_{\text{clean–100}}$.

Figure 3.5 plots quality score against sentence duration. Both systems perform poorer with longer sentences, but the trend is more pronounced for $\text{T2M}_{\text{clean–100}}$ than for T2M_{ADD}. The average number of letters per training sentence for $\text{T2M}_{\text{clean–100}}$ was 100, whereas for T2M_{ADD} it was 117 (averaged across both clean and corrupted utterances). T2M_{ADD} was exposed to longer sentences during its training, potentially equipping it with better generalisation capabilities for handling longer sentences. This observation offers a plausible explanation for the quality distinction between the two systems and underscores the importance of dataset composition when training TTS models.

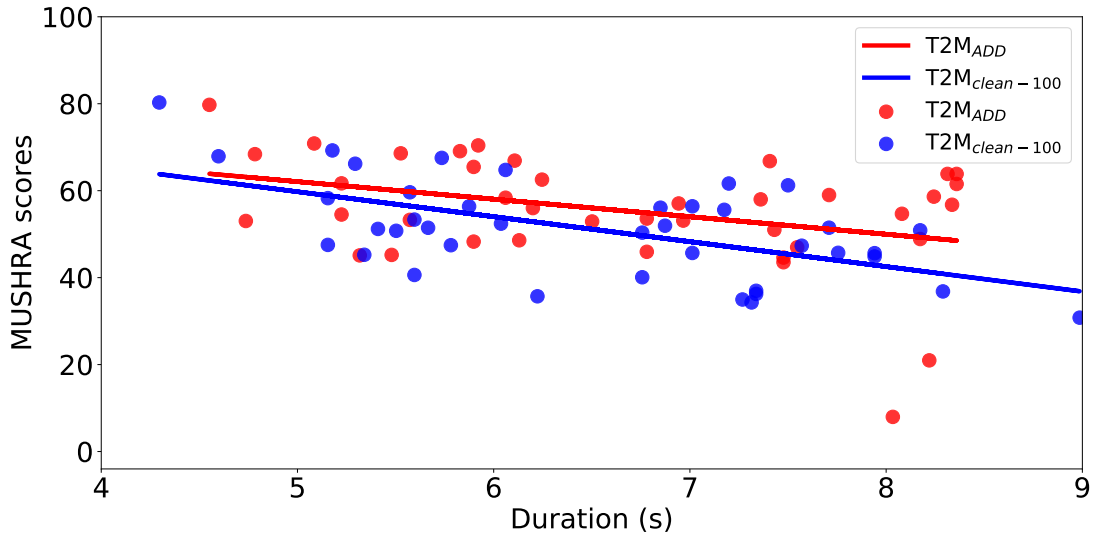

Figure 3.5: MUSHRA score for each listening test evaluated sentence (averaged across listeners) vs. sentence duration. The red and blue lines are a linear fit to the data.

3.6.2 Pronunciation errors

Table 3.2 reports the total number of mispronunciations per system. We identified mispronunciations by listening to every synthesised test utterance. The most common mispronunciations included incorrect vowels, missing nasals, and missing word-initial plosives. The two clean systems and T2M_{ADD} performed reasonably well in this regard. Interestingly, although T2M_{ASR} received low quality ratings in the listening test, it produced fewer pronunciation errors than any other system, partially supporting H_6.

Table 3.2: Number of mispronounced words in the test sentences.

T2M clean–100	T2M clean–50	T2M ADD	T2M ASR	T2M DEL	T2M REPL
45	45	51	34	63	68

3.7 Discussion

In our experiment we chose to corrupt utterances by adding, removing, and replacing words at *random* positions. This could influence the implicit syntactic and semantic analysis performed by the neural front-end. Although we believe our approach does not impact the findings of this chapter or the rationale for future chapters, we think the experiment could be improved by restricting word corruptions to the correct part-of-speech tags based on their position in the utterance.

In evaluating the models discussed in this chapter, we asked participants to rate the stimuli based on overall *quality*. While this measure does not perfectly correlate with accurate or inaccurate pronunciation, which is the primary focus of this thesis, we believe it serves as a reasonable proxy. This limitation is partly attributable to the timing of the research within my PhD, conducted prior to my commitment to specifically investigate and improve phonemic pronunciations in neural TTS using low-resource methods.

In Section 3.5, we observed that $\text{T2M}_{\text{clean–50}}$ outperformed $\text{T2M}_{\text{clean–100}}$. This raises concerns about the appropriateness of the model training and the validity of the results for comparison. Different models may require varying numbers of training epochs, making it potentially misleading to compare them at the same number of epochs. Ideally, each model should be trained to its optimal number of epochs. Nonetheless, these cross-model comparisons do not undermine the primary findings of this chapter: that neural TTS can effectively learn to handle certain types of errors, a capability that prior TTS methods lack by default.

3.8 Conclusion

Our findings suggest that Sequence-to-Sequence (Seq2Seq) Text-to-Speech (TTS) models with an attention mechanism are robust only to *specific* types of transcrip-

tion errors in the training data. They excel at ignoring extraneous information in the input, such as word insertions. However, they struggle with misleading inputs, such as word substitutions, and output that cannot be predicted from the input, like word deletions. While our investigation centred on gross *word-level* errors in the input, the observed robustness of Seq2Seq TTS models extends to finer-grained discrepancies between input and output sequences as evidenced by the competitive performance of the TTS system trained on a transcript corrupted by Automatic Speech Recognition (ASR) errors. It is also reasonable to conjecture that this robustness might extend to other finer-grained inputs such as phonemes. In real-world phoneme transcripts, subtle mismatches can be attributed to various factors, including disfluencies, pronunciation variation due to elision, or speaker accent.

In the next chapter, our attention will shift towards the examination of finer grained mismatches stemming from *pronunciation prediction errors*. These errors are a recurring challenge and can be attributed to various factors, including the use of automated Grapheme-to-Phoneme (G2P) conversion, or incorrect post-lexical processing. These discrepancies often result in phoneme sequences that do not precisely match the speech audio. It is worth noting that input graphemes, in contrast, can *potentially* match more reliably with the speech audio. Consequently, it is possible that training using grapheme inputs may yield a higher-quality voice output, when compared with training using phoneme transcripts of inferior quality. This holds critical significance, particularly in the context of employing TTS for low-resource languages, as grapheme transcriptions are considerably more accessible than phoneme resources.

Chapter 4

Comparing grapheme-input to phoneme-input for text-to-speech

This chapter represents an extended version of my research presented in *A Comparison between Graphemes and Phones as Input to Sequence-to-Sequence Models for Speech Synthesis*, which was published in the proceedings of the Speech Synthesis Workshop 2019 (Fong et al., 2019a). This was joint work done in collaboration with Jason Taylor, Korin Richmond, and Simon King.

4.1 Introduction

In the preceding chapter, we established a key benefit of sequence-to-sequence (Seq2Seq) text-to-speech (TTS) models. We demonstrated that it is possible to train such models on imperfect transcripts without a substantial impact on naturalness. This capability is particularly promising for low-resource applications as it renders training on more widely available found data feasible.

Another democratising benefit of Seq2Seq TTS lies in its capacity to utilise *grapheme* inputs (Shen et al., 2018a). While using grapheme inputs removes the need for phonemic resources, they do present a trade-off – namely, a potential reduction in pronunciation control when the spoken form is not perfectly predictable from graphemes. In contrast, phonemes have seen broader usage in TTS production systems due to their consistent mapping to acoustic features.

Consequently, a question arises: when compared to phoneme-based systems, what

is the *extent* of the impact on pronunciation when training with grapheme inputs? The answer to this question holds significance for making informed decisions when constructing a TTS system. This comparison will offer valuable insights, guiding decisions such as the allocation of resources to phoneme-related components such as lexicons or grapheme-to-phoneme (G2P) models, versus training with only grapheme inputs. In this chapter, we will estimate the effective phoneme error rate for grapheme-based TTS in order to compare it with phoneme-based TTS.

A fundamental task in text-to-speech synthesis is disambiguating complex graphemes-to-speech (G2S) relations. Due to language evolution, the written form is not guaranteed to intuitively map to the spoken form of words. For instance, *'sing'* and *'sine'* only share the sound of their first grapheme *'s'* despite only differing by their final graphemes *'g'* and *'e'*. The standard approach to dealing with this inconsistent mapping in statistical parametric speech synthesis (SPSS) systems has been to employ forced-alignment to sequentially delimit and align segments of speech using linguistic symbols in order to train an acoustic model. The more monotonic and predictable the mapping is between linguistic symbols and speech segments in the training data, the higher the quality of the resulting model. Unlike graphemes, phonemic transcriptions deliberately map to the sounds of a language in a more consistent and perfectly monotonic fashion. Thus, phonemes represent speech more faithfully than graphemes.

Generating phoneme sequences corresponding to input text is then typically handled by a sequence of processing modules chained together, for example in front-end packages such as Festival (Clark et al., 2007b), Mary (Schröder and Trouvain, 2003) or Sparrowhawk (Ebden and Sproat, 2015). The front-end typically performs disambiguation of non-standard words such as numbers, abbreviations and homographs, and post-lexical rules such as the addition of linking or intrusive /r/ between two words, followed by pronunciation dictionary lookup for dealing with in-vocabulary words (IVs), and a G2P model for dealing with out-of-vocabulary words (OOVs). Development of these front-end modules requires a large amount of expertise and effort. This is especially true of building a lexicon. This development cost means that TTS technology is commercially viable only for the world's most widely spoken languages. Although there are over 7000 languages spoken around the world (Ethnologue, 2019), high quality commercial voices exist for just over 30 of them (Amazon, 2019; Google, 2019).

In a bid to avoid such high effort and expense, recent work on neural Seq2Seq models proposes training with grapheme inputs *directly* (Wang et al., 2017a), avoiding the need for some explicit front-end modules. Unlike systems dependent on forced-alignment, Seq2Seq models employ a text encoder with an attention mechanism that learns pronunciation information by considering the entire input sequence at once. This means that the speech can be dynamically predicted from a longer contextual input window than encompassed by incremental HTS-style labels (Zen, 2006) in feedforward DNN-based systems such as Merlin (Wu et al., 2016). This distinction is depicted in Figure 4.1. The pronunciation of grapheme clusters like 'gh' in '*tou**gh***' and '*thou**gh***' are more successfully predicted from graphemes in a Seq2Seq model, as the context of the entire word can be used rather than graphemes consumed incrementally.

However, evidence from multiple studies (Kastner et al., 2019; Shen et al., 2018a; Taylor and Richmond, 2019a) suggests a monolithic Seq2Seq model learns a weaker joint pronunciation and acoustic model from graphemes than from phonemes. First, graphemes in the training data can denote speech sounds ambiguously. For instance, homographs may have distinct pronunciations depending on the surrounding semantic or syntactic context. Additionally, grapheme clusters may be pronounced differently depending on the surrounding graphemic context, for example the bold graphemes in '*tou**gh***', '*w**o**men*' and '*na**ti**on*' represent different sounds from the same graphemes in '*thou**gh***', '*w**o**mb*' and '*na**ti**ve*'. It is clear that graphemes, unlike phonemes, introduce a undesirable level of uncertainty in expected pronunciations. Second, TTS training corpora usually contain fewer wordtypes than a lexicon. This means that the implicit pronunciation model in a grapheme-input Seq2Seq system may be exposed to fewer irregular G2S relations such as in words like *'Flaubert'* or *'baguette'*. Furthermore, words (and their G2S relations) in training corpora for Seq2Seq models are distributed with unbalanced frequencies when compared to lexica used for training G2P models (in which each wordtype occurs exactly once). Rare words, or words with unusual G2S relations, occur far less frequently in Seq2Seq training corpora than common vocabulary and functional words, such as articles (*a*, *the*, etc.) and prepositions (*for*, *on*, etc.).

Each of these factors is likely to contribute to the degradation observed in grapheme-based systems. When building an English Seq2Seq TTS system, we face a dilemma between investing in front-end resources to improve pronunciation or

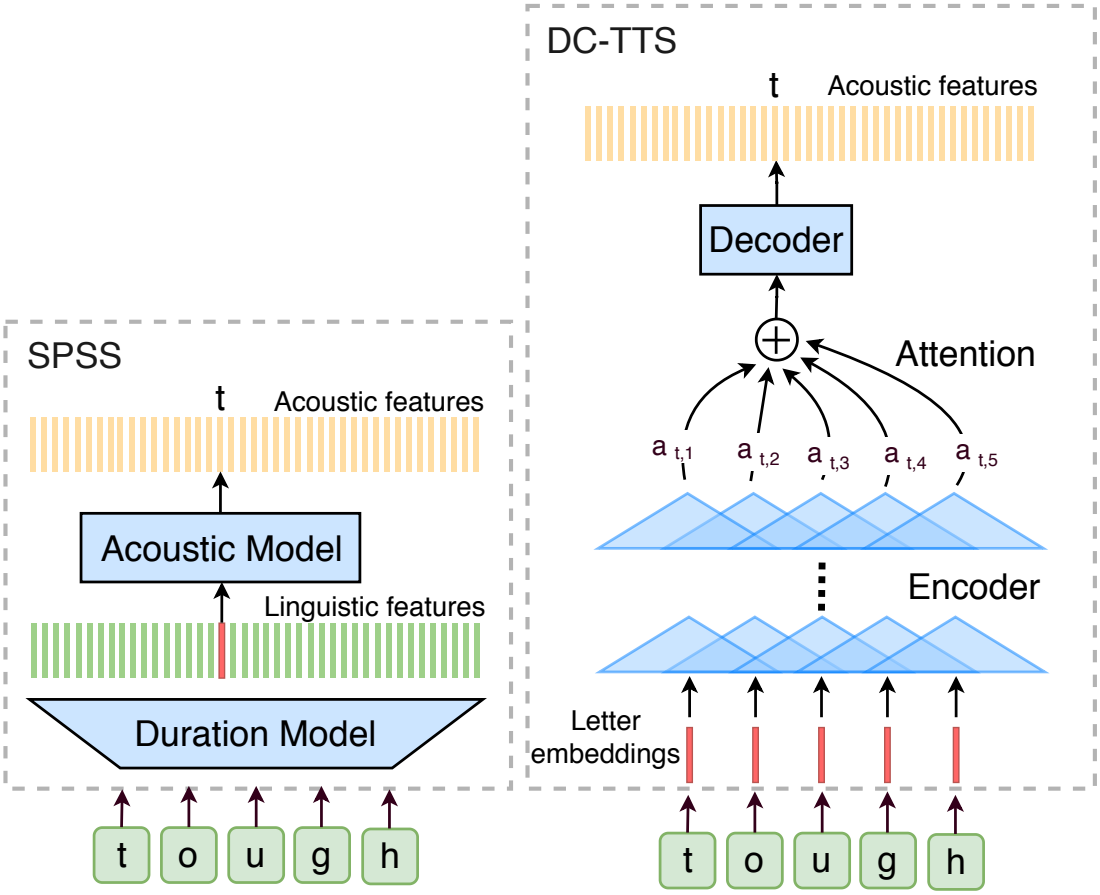

Figure 4.1: An comparison between the context window size that is available to the Acoustic Model in SPSS and the Decoder in DC-TTS. Highlighted in red are the input features available when predicting output acoustic features at a particular timestep 't'. The decoder in DC-TTS has access to *all* the graphemes in the input sequence due to the use of the attention mechanism. Conversely the SPSS acoustic model takes only *one* frame of linguistic features as input. Exactly how many graphemes this spans depends on the nature of the contextual grapheme features it contains.

using graphemes as input and accepting less than optimal pronunciation accuracy.

There has been limited prior work on directly measuring the gap in performance between different input forms for Seq2Seq TTS models. Kastner et al. (2019) compared systems where the input representation was mixed between graphemes and phonemes. Although they did not train systems separately with either graphemes or phonemes alone, they noted that the inclusion of phonemic information during training improved overall voice quality even when using only graphemes at synthesis time. The authors of Tacotron 2, Shen et al. (2018a), noted that acoustic quality was poorer when using graphemes, with mispronunciations and incorrect

prosody found to be among the system's most common errors. This was shown again by Taylor and Richmond (2019a), where words with ambiguous relationships between their orthographic form and phonemic realisation (e.g. foreign names and loan words) were problematic for the pronunciations of a Seq2Seq TTS model.

In this chapter, we aim to build on this prior work by quantifying the overall degradation when training with graphemes instead of phonemic input. We presume that grapheme-input Seq2Seq models must implicitly learn an internal counterpart to G2P conversion and therefore inevitably make errors. Such a model can be viewed as phoneme-input Seq2Seq with inaccurate phoneme input. We compare the grapheme-input model to phoneme-based equivalents with varying proportions of phonemic corruption. By truncating a lexicon and phonemicising an increasingly larger proportion of the training data with G2P modules of varying quality, we obtain systems that simulate the G2S ambiguity that occurs when using graphemes. We then rank the systems by measuring their naturalness in a MUSHRA evaluation.

4.2 Experiments

The primary objective of our experiments is to estimate the pronunciation accuracy of a grapheme-input TTS model. To achieve this, we compare a grapheme-input model against phoneme-input models trained on phoneme transcripts that lie on a spectrum of varying quality. In this work, we used naturalness as a proxy for pronunciation accuracy as, at the time, we were more focused on the holistic impact of how varying inputs to TTS affects its output. Our research questions in this work are:

- How does grapheme-input TTS compare with phoneme-input TTS in terms of overall naturalness?

- How does G2P quality impact the naturalness of phoneme-based TTS?

In the following section we will describe the architecture of the Seq2Seq TTS model, the dataset used for model training, the pronunciation lexicon and G2P models used for phonemicising the dataset, and our subjective listening test evaluation procedure.

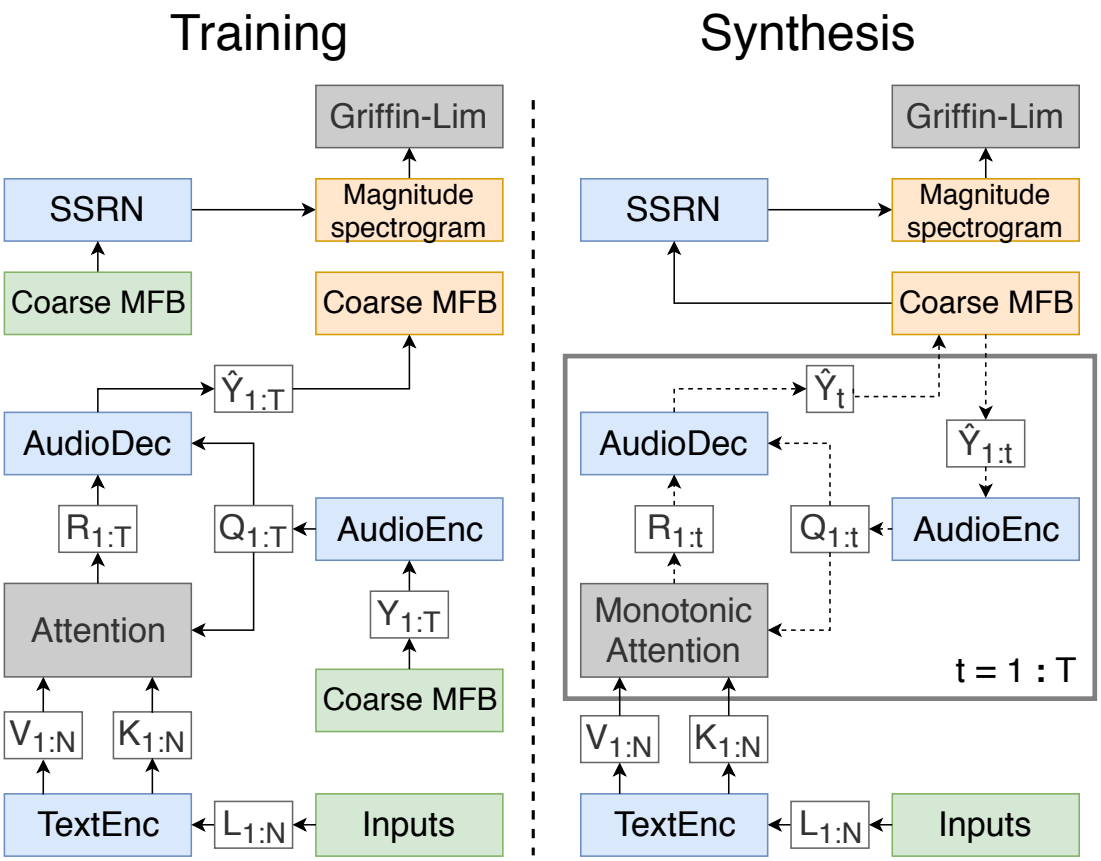

Figure 4.2: Architecture of the Deep-Convolutional Text-to-Speech system (DC–TTS). Components are color-coded as follows: Blue modules are trainable and consist of convolutional layers; Grey elements represent non-trainable computational operations; Green elements signify the input features; Orange elements represent the predictions made by the model. Plate notation over $t = 1 : T$ indicates the autoregressive nature of the decoder, looping from the first to the T^{th} timestep during synthesis. MFB stands for Mel-filterbanks, which were the outputs of the TTS acoustic model, and the inputs to the Super Spectrogram Resolution Network (SSRN).

4.2.1 Text-to-speech Model Architecture

We trained deep convolutional neural sequence-to-sequence TTS models (DC–TTS) (Tachibana et al., 2018c) with either phoneme or grapheme inputs using Ophelia (Watts, 2019). This architecture performs two sequential tasks to produce speech spectrograms from linguistic inputs, as shown in Figure 4.2. First, the Text-to-Mel network (T2M) uses sequence-to-sequence text and audio encoders with attention (Bahdanau et al., 2015) to consume the inputs and predict mel-spectrograms. The Super Spectrogram Resolution Network (SSRN) then consumes these mel-

spectrograms and upsamples them in both time and frequency to produce a full magnitude spectrogram. Finally, the Griffin-Lim algorithm is used to re-introduce phase to the magnitude spectrogram and thus create the output speech waveform. To streamline the training procedure, we adhere to the standard DC–TTS procedure by training the T2M network and the SSRN separately using ground truth data.

All systems use distinct T2M modules trained for 500 epochs on their respective input training data, but share a common SSRN module, also trained for 500 epochs. At test time, each trained system uses the same 0% phoneme word error rate (phoneme WER) test transcript, rather than test transcripts corrupted in the same way as their training transcripts. In the context of G2P prediction errors, we use phoneme WER to quantify the pronunciation accuracy of each TTS training transcript. We define that a phoneme word error occurs when at least one phoneme in a word is incorrectly predicted, according to the pronunciation lexicon. The phoneme WER is then defined as the percentage of phoneme word errors in a transcript. Further details regarding the system architecture and hyperparameter setup are identical to those in the previous chapter and are provided in Section 3.3.4.

4.2.2 Speech Data

We used paired text-audio data from the Linda Johnson (LJ) corpus (Ito and Johnson, 2017) which contains 24 hours of audio distributed over 13,100 utterances taken from 7 non-fiction books. The text was normalised by Jason Taylor using Festival to expand out numbers, ordinals, and monetary amounts. We trained Seq2Seq TTS models using a subset of 9871 utterances from this corpus. Utterances which contained OOVs according to the Combilex pronunciation lexicon (Richmond, 2018) were removed to allow for a training transcript where all words were in-vocabulary (IV). This ensured that our topline model `100combi` (also defined in Table 4.1), is fully phonemicised by Combilex and therefore has a phoneme WER of 0.0%. Furthermore, 242 test utterances (i.e. the in-vocabulary utterances from chapter 50 of the corpus) were held out from training for use in the listening test.

4.2.3 Lexicon and grapheme-to-phoneme models

The General American (GAM) surface-form of the Combilex speech technology lexicon (Richmond, 2018) was used to phonemicise the LJ transcript, and to train a classification and regression tree (CART) as well as neural G2P models. The GAM surface-form is a version of Combilex tailored to typical US speech. Combilex was preferred over the widely used Carnegie Mellon Pronouncing Dictionary (CMU, 2019), for its higher consistency and accuracy. The lexicon and G2P models were integrated into Festival's standard front-end pipeline. The G2P models were used when the GAM surface-form lexicon was truncated (covering only either 50% or 0% of the training transcript) to deliberately induce phone corruption, as explained in Section 4.2.4.

The neural G2P model was a Bidirectional Long Short Term Memory (BiLSTM) network, identical to the one described by Taylor and Richmond (2019a), built by Jason Taylor using the machine translation focused framework OpenNMT implemented in Pytorch (Klein et al., 2017). The encoder and decoder are each composed of 6 bidirectional LSTM layers with 500 hidden units each, and are connected using Luong's global attention (Luong et al., 2015b). The G2P model is trained to predict a sequence of phonemes from a sequence of graphemes using a learning rate of 0.0001, a dropout rate of 0.1, mini-batches of 64 samples, and was optimised using the Adam optimiser (Kingma and Ba, 2015). The BiLSTM G2P model converged after 50,000 training steps.

The CART G2P model was built using Festival, based on the system described in Richmond et al. (2009). This model was trained with the phoneme-grapheme alignment information already provided in Combilex.

4.2.4 Creation of training transcripts

We generate a range of phonemic transcripts with increasing proportions of incorrectly predicted phonemes to serve as a proxy for assessing the errors likely to arise when using graphemic input. Figure 4.3 shows how grapheme input is ambiguous in its relations to speech and how the ambiguity may be approximated by using incorrectly predicted phoneme sequences. We see that the grapheme cluster 'gh' may represent either silence or a voiceless labio-dental fricative.

We create phone transcripts using the following methodology. First, we generate

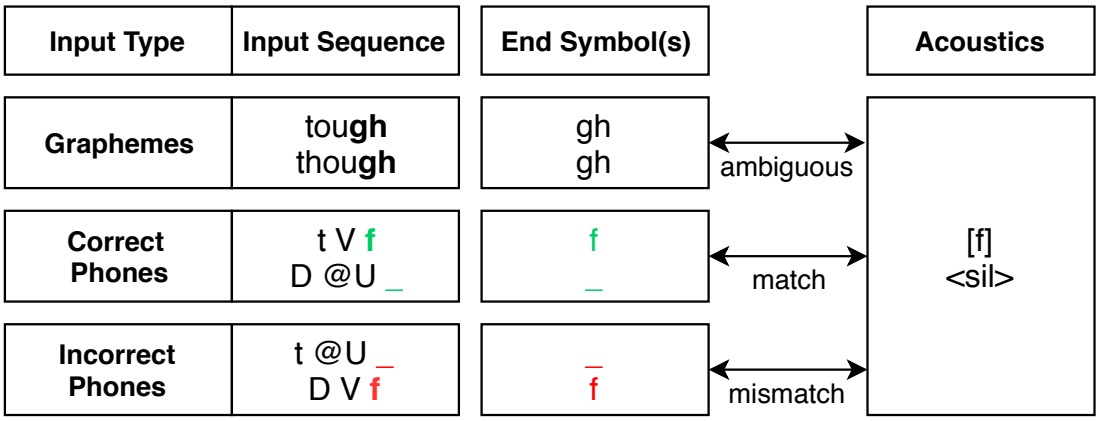

Figure 4.3: Nature of mapping from various input grapheme sequences to acoustics. The underscore symbol '_' represents a missing phoneme from the prediction, and '<sil>' represents a missing sound from the acoustics. Using correctly predicted phonemes results in a match between inputs and acoustics during training that should produce a high quality acoustic model at test time. However using incorrectly predicted phonemes results in a mismatch that may negatively impact the performance of the acoustic model. As the mapping between graphemes and acoustics can be ambiguous given the semantic, syntactic, or graphemic context, we would also expect the use of graphemes to have a negative effect on the acoustic model. The IPA pronunciations for 'tough' and 'though' are /tʌf/ and /ðəʊ/ respectively.

a 0% phoneme WER transcript via look-up using the full lexicon (Note, our measure of phoneme WER does not account for phoneme errors due to context, e.g. lexical stress errors in English). This is as close as we can get to a linguist's transcription given the large dataset sizes used in Seq2Seq TTS. Second, we generate inferior phonemic transcriptions by varying the proportion of the training text phonemicised by lexicon or by differing G2P models. In this way we compare the grapheme-based system to phoneme-based systems with varying consistency in the phonemes-to-acoustics relations.

Table 4.1 shows the breakdown of input data used to train each system. The **Ratio** column details the proportion of each input type to each system. The **PWER** column shows the phoneme WER of each training transcript. The error rates reported for the G2P models were much lower in their respective papers (in which the CART and neural models both scored <15% phoneme WER): this is because those error rates are calculated over their **test** sets (which are held out wordtype-to-phonemes entries from their pronunciation lexica), which do not

Table 4.1: Description of all systems trained for our experiment. The **Input** column denotes the method used to phonemicise the transcript. The **Ratio** column denotes the percentage of the full lexicon used, or the entries replaced by G2P predictions. The Phoneme **PWER** is calculated as the percentage of words containing a phoneme error in the transcript.

Name	Input	Ratio (%)	PWER (%)
`100combi`	Lexicon Lookup (LL)	100	0.0
`50neur`	LL / Neural G2P	50 / 50	11.5
`50cart`	LL / CART G2P	50 / 50	14.3
`100neur`	Neural G2P	100	25.2
`100cart`	CART G2P	100	30.6
`let`	Graphemes	100	-

reflect the natural frequencies of wordtypes in our **training** LJ speech transcript. This difference is an important consideration when integrating a G2P model into a developed TTS system, as the phoneme WER of a test set may not reflect the phoneme WER on natural occurrences of words as employed in the TTS training data domain.

Importantly, some G2P errors may in fact be plausible variants, such as a prediction of the word 'tamil' with [I] instead of a stated schwa phone [@] in the lexicon. This is not a gross error that would degrade TTS quality noticeably. The extent of this effect is however left unquantified and has not been measured as it would require a manual review of all errors in the training transcripts.

4.2.5 Listening test

The six systems in Table 4.1 were submitted to a MUSHRA test with natural recordings (generated by copy synthesis using Griffin-Lim) as the hidden reference. We recruited 30 English native speakers as listeners. The listening tests were conducted in purpose-built listening booths.

The focus of our inquiry was the training transcript and its effect on the quality of a built voice. Hence, only words with mappable pronunciations (i.e. easily predictable from graphemes) were used at test time. We hand-selected 20 ut-

terances of such words from our 242 test utterances. These utterances did not require any disambiguation via the traditional front-end because homographs and abbreviations were excluded and numbers verbalised. In this way, any errors resulting from grapheme ambiguity in test utterance words were minimised. All listeners rated all 20 utterances. For a fair comparison of the phoneme-based TTS models, we used the same Combilex transcriptions from the complete lexicon at test time.

4.3 Results

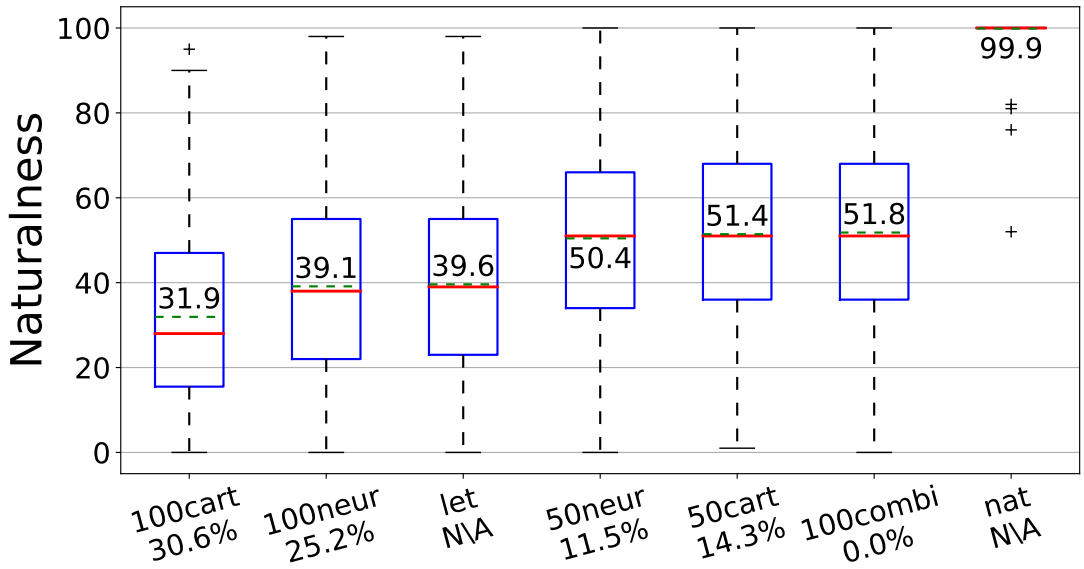

Figure 4.4: MUSHRA results from the subjective listening tests. Solid red lines are **medians**, dashed green lines are **means** (also numerically labelled), blue boxes show the **25th** and **75th** **percentiles**, and whiskers show the **range** of the ratings, excluding **outliers** which are plotted with +. Percentages below system names indicate the phoneme WER of their respective training transcript.

4.3.1 System comparison

4.3.1.1 Overall trends

The results of the MUSHRA test are displayed in Figure 4.4. Participants were instructed to raise the score of the highest quality voice (i.e. the copy synthesised natural hidden reference) to 100, which is evident in the results. No

such stipulations were made for other voices, though, and all TTS systems scored on average below 52 on the naturalness scale, including the system trained with a 0% phoneme WER transcript. We did not exclude any participants from the experiment, as none of them consistently rated the hidden reference at a score different from 100.

Whilst we evaluated the general performance of each model by varying the training data of each T2M network, artefacts resulting from the use of the Griffin-Lim algorithm are likely to have influenced the average score for each system, potentially masking the finer differences between each T2M network. While the scale we used for this experiment only measured naturalness, we hypothesise that intelligibility may be affected by Griffim-Lim distortion. If a higher quality replacement of Griffim-Lim were used to re-introduce phase, analysing whether this hypothesis is true would be unnecessary. Therefore, in subsequent chapters, we replace Griffim-Lim with a neural vocoder.

4.3.1.2 Acoustic model not significantly degraded when phonemicising 50% of the transcript using G2P

The differences between the three best performing systems `100combi`, `50cart`, and `50neur` are negligible and not statistically significant. While this equivalence could be interpreted as suggesting that training transcripts with a phoneme WER corruption of up to 15% bear negligible degradation generally, in reality a large proportion of the words in fact have viable pronunciations. That is, incorrectly predicted phonemes could still be acoustically similar to the corresponding speech data and thus will not *greatly* degrade the acoustic model. Unfortunately, teasing apart which predictions are viable and which are implausible (i.e. would have a larger detrimental effect in training) is non-trivial without manual verification. Furthermore, the differences between these systems may also be caused by other factors such as Griffim-Lim or the particular random seed that initialises the parameters of DC-TTS during training.

4.3.1.3 Using graphemes is equivalent to phonemicising using only neural G2P

There is a 23.5% relative drop in the naturalness score in the Seq2Seq model when trained on graphemes (`let`) rather than phonemes from the full lexicon

(100combi), from 51.8 to 39.6. The MUSHRA scores are significantly different at $p < 0.0005$. In terms of phonemic corruption meanwhile, the naturalness of let is not significantly different to that of a phoneme-based system trained with 25% phoneme WER (100neur) which has a naturalness rating of 39.1.

We also observed from our subjective listening that the grapheme-based system let, as well as 100cart and 100neur synthesise speech less 'crisply'. Furthermore, there are noticeable pronunciation errors in their synthesised test utterances, unlike in the other systems (even though correct phonemes are being used at test time). Examples of pronunciation errors may be heard on the samples page [1]. This demonstrates that ambiguous and indirect input-to-acoustics mappings at training time may lead to demonstrable degradation on acoustic quality at test time.

4.3.2 Out of vocabulary words in phoneme-based systems

We excluded utterances which contained OOVs from the training set. This was to ensure that the lexicon contained pronunciations for all remaining word tokens when building our topline model 100combi. While it is inherently difficult to measure the effect of using G2P to predict pronunciations for error-prone OOVs, the relatively strong performance of 50neur and 50cart suggest that OOVs in large datasets could generally be phonemicised by a G2P model without a severe impact on the general acoustic quality of the TTS model.

As Table 4.2 shows, although approximately a quarter of utterances in the LJ dataset contain an OOV, <2% of total tokens are OOV. Even though 50neur and 50cart possess a phoneme WER of up to 14.3% calculated across all tokens, they still rendered similar performance to the TTS system trained with all tokens being IV and phonemicised exclusively by lexicon lookup (100combi). Even if all OOV tokens were mis-phonemicised in LJ, and their respective utterances were included in the TTS training data, we believe that the overall effect on TTS performance would be negligible.

Table 4.2 also shows how OOVs are distributed in other corpora used for training TTS models. Even in a more phonemically diverse corpus such as Nancy, from the Blizzard Challenge 2011 (CSTR, 2011), or a large corpus of audiobooks like

[1] https://jonojace.github.io/SSW19-comparison

LibriTTS (Zen et al., 2019), OOV token rates of 4.9% and 1.3% respectively suggest that using a G2P model for these OOV tokens during training would not be significantly detrimental to the acoustic model of a phoneme-based TTS system. In addition, the full dataset may then be utilised in training, compared with training with only IV utterances.

Table 4.2: Combilex OOVs in large TTS datasets. The **type** and **token** rates describe how many individual wordtypes and tokens are OOV. The **utt** rate is the percentage of utterances containing at least one OOV token.

Dataset	Hours	OOV Type Rate (%)	OOV Token Rate (%)	OOV Utt Rate (%)
LJ	24	9.8	1.9	24.6
Nancy	17	10.5	4.9	56.9
LibriTTS	585	32.8	1.3	18.0

4.4 Discussion

Similar to Section 3.5 and what was discussed in Section 3.7, the experiment in this chapter involves training models for the same number of epochs, rather than optimising the number of epochs for each model individually. Although this approach is not ideal, we contend that its impact is mitigated by the fact that each model was trained using the same number of utterances.

4.5 Conclusions

Graphemes in English have a more ambiguous relationship with acoustics than phonemes. While text encoders and decoders with attention mechanisms in Seq2Seq TTS models lend themselves to coping with the non-monotonic and unpredictable nature of graphemes-to-speech relations in English, training with grapheme input still does not work as well as phoneme input.

In this chapter, we sought to quantify the gap in performance between grapheme-based and phoneme-based Seq2Seq TTS models by performing a MUSHRA evaluation with phoneme-based systems trained with differing amounts of incorrect

phonemes as generated by standard G2P models. We found that the grapheme-based system performed worse than a phoneme-based system with 0% phoneme WER training transcript, and roughly equivalent to a system with a 25% phoneme WER. Furthermore a system with a <15% phoneme WER training transcript performed with negligible differences to a 0% phoneme WER one, suggesting that OOV tokens in training transcripts may be phonemicised by a G2P model without significant impact on the TTS acoustic model. This potentially removes the need to manually add OOVs to the lexicon when incorporating more data into the training set.

In the next chapter, motivated by the results that suggest that the *quality* and *control* of pronunciations is impacted when using graphemes, we investigate whether pronunciation control can be maintained when using a *smaller* (and thus cheaper) pronunciation lexicon which can, consequently, only phonemicise a smaller proportion of the training transcript. In the experiments of the next chapter, we simulate smaller lexicons by restricting the number of wordtypes that they contain. IV words are phonemicised by lexicon lookup, and OOVs are left as grapheme sequences during training. Training using a mixture of graphemes and phonemes is possible due to the use of the representation mixing paradigm in the next chapter (Kastner et al., 2019), where a TTS model trains on utterances where words are represented by either grapheme or phoneme sequences randomly, which enables the model to use either graphemes or phonemes at synthesis time.

Chapter 5

Grapheme-phoneme representation mixing with partial coverage pronunciation lexica

This chapter represents an extended version of my research presented in *Testing the Limits of Representation Mixing for Pronunciation Correction in End-to-End Speech Synthesis*, which was published in the proceedings of Interspeech 2020 (Fong et al., 2020). This was joint work done in collaboration with Jason Taylor and Simon King.

5.1 Introduction

In the previous chapter, we showed that there is an increase in naturalness when training a TTS model using phoneme inputs rather than grapheme ones. However, the use of phoneme inputs incurs the costs of a pronunciation lexicon and grapheme-to-phoneme (G2P) model which are both required to *fully* phonemicise a transcript.

With the goal of alleviating this cost, in this chapter, we investigate whether TTS pronunciations remain accurate when using *smaller* amounts of phoneme resources that can only *partially* phonemicise a transcript. If this were indeed the case, then it would be possible to build a TTS system with controllable pronunciations in lower resource settings.

The ability to control pronunciation is necessary for text-to-speech synthesis (TTS) in deployment. Broadly speaking, non-phonemic orthographies (as in English) have an ambiguous relationship to acoustics and cause pronunciation prediction to be unreliable. Traditionally, the ambiguities between graphemes and acoustics are tackled by creating a linguistic specification in the front-end. The linguistic specification is an intermediate representation between graphemes and acoustics that contains various levels of linguistic features (e.g. phonemic, syllabic, syntactic, prosodic) to improve pronunciation control. Examples of front-end packages that can generate such linguistic representations include Festival (Clark et al., 2007b), Mary (Schröder and Trouvain, 2003) and Sparrowhawk (Ebden and Sproat, 2015). However, developing front-end modules is expensive, because creating entries in a pronunciation lexicon requires manual transcription by trained linguists. One such example is Unisyn, which took an estimated 2 person-years to create (Fitt, 2001). Other available pronunciation lexica for TTS include CMUdict (CMU, 2019) and Combilex (Fitt and Richmond, 2006).

The recent application of neural sequence-to-sequence modelling (Seq2Seq) to TTS has enabled an *end-to-end* (E2E) paradigm. This is where speech is predicted directly from graphemes without the use of a front-end or linguistic specification. Examples of such models include Tacotron (Wang et al., 2017a) and Tacotron 2 (Shen et al., 2018a). The prospect of TTS without a front-end has led to a growing interest in the E2E paradigm. However, the need to control the pronunciations of output speech runs counter to the current drive towards E2E TTS. A degree of intervention is necessary to control pronunciation but should be kept to a minimum.

Motivated by the need to correct pronunciation in E2E TTS systems, *representation mixing* involves training on a mixture of graphemes and phonemes, with each input word represented either graphemically or phonemically (Ping et al., 2018; Kastner et al., 2019). This induces the model to be able to use any combination of the two modalities at test time. This enables a mostly grapheme-input TTS system to use, when desired, phoneme sequences to represent words in an utterance, without necessitating the use of phonemes for *all* words in an utterance. With the option of using phoneme input, it becomes possible to control pronunciations at test time without the need for a complete phonemic lexicon of all of the word types (in the training data), or a G2P model. However, previous research on

representation mixing has not empirically studied the robustness of phoneme correction, nor the quantity of phoneme labelling required. Their experiments use a large phoneme-based pronunciation lexicon to phonemicise 100% of the word types in the training transcript.

In contrast, in this chapter, we seek to quantify what the minimal phonemic intervention during training should be for a working pronunciation corrector in a Seq2Seq TTS model. We identify and investigate 3 factors relevant to pronunciation correction in the representation mixing paradigm: the number of word types in the training data that are phonemically transcribed, whether these word types are selected according to their token frequency, or whether coverage-based selection algorithms can reduce the amount of phoneme labelling needed.

We train TTS systems using progressively smaller pronunciation lexica and evaluate the accuracy of their pronunciations by expert listening. From our results, we find that from as little as 500 word types, pronunciation correction is possible. We also find that as the number of word types labelled during training increases, the corrective ability improves. However, pronunciation correction equivalent to a highly resourced phoneme-based TTS system is *not* possible under representation mixing. We believe that our results impact the field of under resourced TTS. They suggest that enabling a system to use phoneme pronunciations to control the output of a TTS system *doesn't* require phonemicising the entire transcript during training. Instead, it is possible to enable this control using much smaller lexica which are more feasible to transcribe in real world under resourced situations.

5.2 Experiments

The goal of our experiments was to determine whether TTS pronunciations remain accurate when using smaller pronunciation lexica. To achieve this, we created pronunciation lexica of various sizes and then used them to train a suite of grapheme-phoneme representation mixing TTS models. We then evaluated their ability to correct pronunciation by using these models to synthesise test utterances containing the correct phoneme sequences of words which were mispronounced when using grapheme inputs, and then measured their pronunciation accuracy.

We also compared various methods of choosing which words in the training

data should be phonemically transcribed, for example choosing words based on frequency, or choosing words based on phonemic coverage. This was done in order to inform best practices when creating a lexicon from scratch for the purpose of building a TTS system.

In the rest of this section we elaborate on the speech corpus used to train the TTS models, the lexicon used to create smaller lexica, the methods used to create each smaller lexicon, how we trained TTS models using grapheme-phoneme representation mixing, and our subjective listening tests for evaluation.

5.2.1 Linguistic data

We trained all models using the LJ Speech corpus (Ito and Johnson, 2017) comprising approximately 24 h of audio from 13,100 sentences from 7 non-fiction LibriVox books read by a single American female speaker. The transcript has normalisation performed on numbers, ordinals, abbreviations and monetary units, and we additionally removed capitalisation.

Training a representation mixing model involves randomly replacing a word's graphemes with its phoneme sequence. We obtain phoneme sequences from the Unilex pronunciation lexicon (Fitt, 2001), as it has wider word type coverage (167,000 entries) and better consistency than open source lexica such as CMUdict (CMU, 2019). Unilex uses the Unisyn set of 56 phones (55 of which are found within LJ Speech), and includes both syllable stress and syllable boundary information for each entry, e.g. the entry for 'speechless' is /s p ii ch 1 | lw @ s 0/, where digits encode syllable stress and the '|' symbol represents syllable boundaries.

5.2.2 Resource-limited lexica

As noted earlier, a large lexicon is very expensive to create. To investigate the limits of how well representation mixing can perform pronunciation correction, we designed a range of smaller pronunciation lexica differing in the number of entries, and differing in how the entries were chosen. Only the word types contained in a given lexicon are ever randomly phonemicised during TTS training. Three reference lexica were designed as follows:

- `grapheme-only`: an empty lexicon; training a representation mixing model with this lexicon is equivalent to training with grapheme-only input. This

model was used to determine which words are mispronounced by a typical E2E TTS system.

- `oracle-14`: contains 14 word types, which is the smallest possible lexicon that covers all 55 Unisyn phones that occur in LJ Speech at least once. We use this lexicon to discover whether full phoneme coverage is sufficient to enable pronunciation control.
- `full-13049`: all 13,049 word types that co-occur in LJ Speech and Unilex. This lexicon should have the best possible pronunciation correction ability.

We devised 5 word type selection methods that each lead to a lexicon of n entries. We compare models trained with these lexica to determine the most effective size and contents for a resource-limited lexicon. n can range from 0 to N (here, $N = 13049$ which is the total number of word types that co-occur in LJ Speech and Unilex), and for any $n_2 > n_1$ the lexicon of size n_1 is a strict subset of the lexicon of size n_2. These selection methods are attempting to solve the optimisation problem of choosing the wordtype that maximises $score_i$ at each iteration step under the constraint of choosing n number of wordtypes.:

- `rand-n`: randomly selects n word types. These lexica help us understand whether a naive selection method can outperform more well-reasoned approaches.
- `freq-n`: selects the top n most frequently occurring word types in LJ Speech. These lexica help answer whether choosing to phonemise words that cover the most tokens during training is the best approach.
- `phone-n`: greedily selects n frequently occurring word types while also trying to achieve wide phoneme coverage. We build these lexica using the greedy Algorithm 1 where the units of concern are phoneme unigrams. This is an oracle condition because it uses pronunciation information from `full-13049`.
- `bigram-n`: identical to `phone-n` but uses character bigrams as the units of concern. These lexica help answer whether wide coverage over graphemic contexts is a potential proxy for wide phoneme coverage and thus may result in better pronunciation performance. This is a realistic condition to contrast with the previous oracle condition `phone-n`.
- `trigram-n`: identical to `bigram-n` but uses character trigrams as the units

of concern.

Algorithm 1 Greedily builds a lexicon containing n word types by considering each word type's frequency and linguistic unit coverage. This algorithm is used to build the `phone-n`, `bigram-n`, and `trigram-n` lexica

```
entries ← empty_set()
candidate_words ← all_word types_in_lexicon()
unseen_units ← all_units()
while len(entries) < n do
    while len(unseen_units) > 0 do
        num_unseen_units ← len(unseen_units)
        scores ← score(candidate_words, unseen_units)
        best_word ← max(candidate_words, scores)
        entries.add(best_word)
        candidate_words.remove(best_word)
        unseen_units.remove(get_units(best_word))
        if len(entries) = n then
            return entries
        if len(unseen_units) = num_unseen_units then
            unseen_units ← all_units()
    unseen_units ← all_units()
return entries
```

Alongside Algorithm 1, we use a simple score defined in Equation 5.1 to build the coverage-based `phone-n`, `bigram-n` and `trigram-n` lexica. The score weights between frequency and linguistic unit coverage. The score for word type w_i is the frequency of w_i in the speech corpus multiplied by the number of its units that overlap with the current set of unseen units. Using this score within Algorithm 1 favours adding words that cover unseen units.

$$score_i = freq_i \times (1 + num\_overlapping\_units) \tag{5.1}$$

5.2.3 Models

We closely followed the representation mixing training approach detailed by Kastner et al. (2019). During training, the graphemes of any word in the input

can be randomly replaced by its phoneme sequence. This occurs at a fixed mixture probability of p_{mix} *only* for the words present in the lexicon being used. In all experiments, we used the same $p_{mix} = 0.5$ as reported by Kastner et al. (2019). We implemented representation mixing training with a Tacotron 2 (Shen et al., 2018a) open source implementation (McCarthy, 2019). The Tacotron 2 model predicts mel-spectrogram frames, from which we use WaveRNN (Kalchbrenner et al., 2018) (a single model trained on the LJ Speech corpus is used in all systems) to generate waveforms. Each TTS model was trained on a single Nvidia GTX 1080, with a batch size of 32, learning rate of 0.001 and 350,000 total training steps stopping criterion.

We trained one model for each reference lexicon described in Section 5.2.2 – `grapheme-only`, `oracle-14`, `full-13049` – and one model for each combination of word type selection method and value of n in $\{500, 2000, 4000, 6000\}$, for a total of 23 models.

Furthermore 5 supplemental models were trained using the `full-13049` lexicon to answer further questions not related to lexicon size or composition:

- `mixprob-up` and `mixprob-down`: linearly vary p_{mix} so that it depends on the frequency rank of the word. `mixprob-up` phonemicises the most common word in LJ Speech with $p_{mix} = 0.5$, and the least common word with $p_{mix} = 0.9$. These values are swapped for `mixprob-down`. These models help investigate whether phonemicising the most frequent or the least frequent words more often during training can be beneficial for pronunciation correction.

- `stress`, `syllable`, and `stress-syllable`: these models additionally include a word's stress and/or syllable information when it is phonemicised during training. These models help investigate whether additional linguistic markup aids pronunciation correction.

After training all the models, we then generated our 3 sets of test stimuli[1]. Table 5.1 shows how many tokens in LJ Speech are covered by each lexicon (expressed as a percentage of the tokens covered by `full-13049`) and therefore could be randomly phonemicised during training.

[1]Samples available at `jonojace.github.io/IS20-repmixing-limits`

Table 5.1: Number of word tokens in LJ Speech covered by each resource-limited lexicon expressed as a percentage of the 223179 tokens covered by `full-13049`. Additionally: `oracle-14` covers 41 tokens (0.018% of `full-13049`).

n	500	2000	4000	6000
`rand`	3%	19%	43%	56%
`freq`	69%	86%	93%	96%
`bigram`	55%	75%	85%	90%
`trigram`	44%	49%	65%	72%
`phone`	66%	81%	90%	94%

5.2.4 Evaluation

We developed 3 sets of words, each carried in the sentence "Now we will say ... again." as test stimuli. Each set helps answer a specific question regarding all models:

In-LJ: Phonemic sequences for 50 words that occur in LJ Speech but were never phonemicised during training (i.e. by chance, they were always represented as graphemes), and are mispronounced by the baseline `grapheme-only` model: despite being in the training data, they are surprisingly still mispronounced when represented as graphemes. This set investigates which models offer effective phoneme correction for this category of unexpected mispronunciations.

Out-LJ: Phonemic sequences for 50 words that do *not* occur in LJ Speech, and are mispronounced by the baseline `grapheme-only` model. These represent the key challenge of generalising to words without spoken examples.

Phonemicised: Grapheme sequences for 50 words that occur in LJ Speech, *were* phonemicised, and are pronounced *correctly* by the baseline `grapheme-only` model. This test set checks that representation-mixed training preserves correct output from the grapheme-only case.

Judgements of pronunciation correctness require careful listening because errors are not simply a matter of a categorically-different phone being produced but can be subtle and ambiguous. Therefore, we use expert listeners. We obfuscated the above stimuli and presented them in random order to two expert listeners (myself and Jason Taylor) who independently judged whether each test word

was pronounced correctly. The listeners had available to them the intended pronunciation for each stimulus. In cases of disagreement, they discussed and re-listened to reach an agreement. In total each listener judged 2100 stimuli, which took roughly three hours of work.

5.3 Results

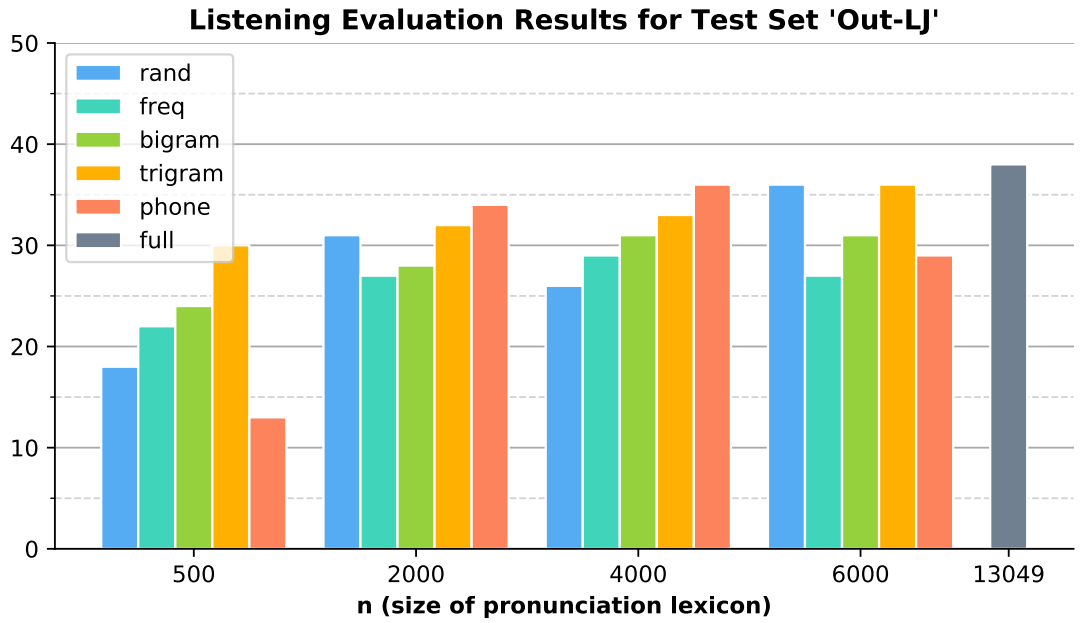

Figure 5.1: Listening test results of models trained using resource-limited lexica generated by the word type selection methods. **Out-LJ** is a test set of 50 words that do not occur in LJ Speech, and are mispronounced by the baseline grapheme-only model. The test set words are represented by correct phoneme sequences in the inputs to the TTS model.

5.3.1 Overall observations

Figures 5.2 and 5.1 visualise the experimental results. First and foremost we observe that 100% correct pronunciation control is not yet possible with representation mixing models. Even the topline lexicon, `full-13049`, achieves a far from perfect score of 38/50 over **Out-LJ**. Although we do not have results for such a model, we believe that a model trained using only phoneme inputs would achieve a higher score since it would not need to use model capacity to learn a mapping between graphemes and speech.

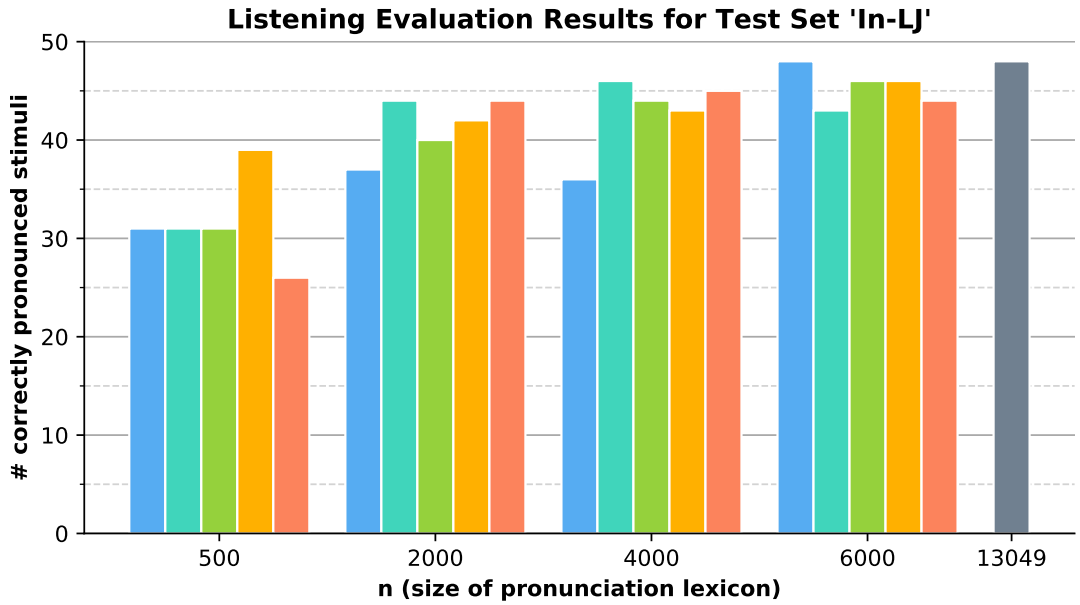

Figure 5.2: Listening test results for models trained using resource-limited lexica generated by the word type selection methods. **In-LJ** is a test set of 50 words that occur in LJ Speech but were never phonemicised during training, and are mispronounced by the baseline `grapheme-only` model. 50 is the *maximum* number of correctly pronounced stimuli that each model can generate. The test set words are represented by correct phoneme sequences in the inputs to the TTS model. This figure shares the same legend as Figure 5.1

Our second observation is that the number of *word tokens* phonemicised during training is not the sole factor that determines pronunciation correctness, especially for the most important test set: **Out-LJ**. In other words, the `freq-n` lexica, which cover the most tokens for any given lexicon size n (see Table 5.1 for word token coverages), do **not** lead to the best pronunciation correction. Rather, the *choice* of word types in the lexicon matters more.

The results in Figures 5.2 and 5.1 also show a trend across all lexica that increasing their size n results in better performance over both test sets, **In-LJ** and **Out-LJ**, which is likely to be simply a result of a greater number and variety of phonemicised word tokens mixed in during training. The results for `oracle-14` support this hypothesis: `oracle-14` phonemicises only 41 tokens during training, and is unable to perform pronunciation control to *any* degree, with scores of 0 on both **In-LJ** and **Out-LJ** (not shown in Figures 5.2 and 5.1) with even the 14 word types seen during training pronounced incorrectly. It is not sufficient to see each phoneme

just a few times in limited contexts to achieve control over pronunciation.

There are a few exceptions to this trend. The performance of `rand-n`, `freq-n`, and `phone-n` can *fall* as n grows: performance from `rand-2000` to `rand-4000`, `freq-4000` to `freq-6000`, and `phone-4000` to `phone-6000`, demonstrate this. Naively using more resources to increase the size of the lexicon doesn't necessarily result in better pronunciation performance. `bigram-n` and `trigram-n` do not exhibit this behaviour, which suggests that the lexicon should contain word types that cover a wide range of grapheme-phoneme contexts. Both `bigram-n` and `trigram-n` – which are both possible in a real application – do this nearly as well as `phone-n` (and actually outperform when $n = 6000$) – which, as noted earlier, is an oracle setting.

As expected, performance on **In-LJ** (words that occur in the LJ Speech training audio) is better than on **Out-LJ**.

5.3.2 Comparison of lexicon selection methods

`rand-n` occasionally outperforms other methods despite covering far fewer tokens in LJ Speech (refer to Table 5.1 for word token coverages): compare `rand-500` with `phone-500`, or compare `rand-6000` with all other $n = 6000$ models. This could be due to `rand` accidentally providing beneficial phonemic coverage, or exposing the neural front-end to less common and potentially more varied combinations of letters. Generally however, `rand-n` performs worse than the other methods when $n \leq 4000$, this along with the fact that `rand-n` isn't a principled or predictable word type selection method means we cannot recommend it over the other methods, especially when resources are scarce.

`phone-n` performs the best for n of either 2000 or 4000, suggesting that a lexicon with balanced phoneme coverage is important. `phone-6000` and `phone-500`, however, both have surprisingly poor performance. A potential reason could be that Algorithm 1 takes only 41 words to cover all phoneme unigrams once, 313 words to cover all grapheme bigrams once, and 2196 words to cover all grapheme trigrams once. Thus, choosing word types by bigram or trigram coverage will tend to select word types with a greater variety of spellings while caring less about token frequency than when covering phoneme unigrams. It is possible that greater spelling variety will correspond to greater phonemic context variety, which may

be better for **Out-LJ** performance. Since `phone-n` is an oracle setting in which an existing large phonemic lexicon is required, it is not a method that would be used in practice in a limited-resource scenario, and thus it is only intended to provide points of comparison with the other methods.

`freq-n`, `bigram-n` and `trigram-n` are, in contrast, all feasible in a real limited-resource use case. The simplest method `freq-n` offers poor performance for **Out-LJ** for all n. `trigram-n` almost always outperforms `bigram-n`. The clear conclusion is that a greater variety of grapheme sequences (i.e., spellings) in the lexicon leads to better pronunciation correction performance, as spelling variety likely correlates with phoneme context variety.

In summary, considering all of the above results, when resources are limited the `trigram-n` lexicon selection method seems to be the best one to use. It offers better pronunciation control than any other method for a lexicon with 500 entries (`trigram-500`), falling not far short of the `full-13049` lexicon, despite being less than 4% of the size (i.e., 25 times cheaper to create, excluding fixed costs).

5.3.3 Additional experiments

The results above indicate that `trigram-n` is the recommended method for creating a lexicon from scratch. In additional experiments, we used the supplemental models listed in Section 5.2.3 to explore two further questions.

5.3.3.1 Phonemicising lower frequency words with a higher probability improves pronunciations

Results for `mixprob-up` (**In-LJ**:47/50, **Out-LJ**:42/50) demonstrate that phonemicising lower frequency words with a higher p_{mix} during training slightly improves pronunciation control for **Out-LJ** words, compared to the uniform $p_{mix} = 0.5$ across all word types used in `full-13049`, which achieves a score of 38/50. We presume this is because low-frequency words will tend to have more variety in their spelling and/or phoneme sequences, which we know to be beneficial from the results for `trigram-n` and `phone-n` respectively. Results for `mixprob-down` (**In-LJ**:48/50, **Out-LJ**:39/50) are very similar to `full-13049`: phonemicising higher frequency words more often has no benefit.

5.3.3.2 Syllable information improves pronunciations

`syllable` is the best-performing across all results (**In-LJ**:47/50, **Out-LJ**:48/50), correctly pronouncing 10 more OOV words than the topline reference `full-13049`. To understand the types of words that `syllable` was able to pronounce correctly *because* it exploits syllable boundary information, we analysed the 7 words (input as phoneme sequences) that `syllable` pronounces correctly with syllable information, `full-13049` pronounces incorrectly, and `syllable` pronounces incorrectly when syllable information is absent. These words are almost all morphological compounds containing a stop (e.g., /t/) and then /h/ occurring across the syllable boundary: goatherd /g ou t | h @@r r d/, loophole /l uu p | h ou lw/, upheld /uh p | h e lw d/, coathanger /k ou t | h a ng | @r r/, plothole /p l aa t | h ou lw/, plughole /p l uh g | h ou lw/, funghi /f uh ng | g ii/. When these words are pronounced incorrectly by `full-13049`, the error occurs over the syllable boundary. For example /l uu p h ou lw/ is mispronounced as /l uu f ou lw/. Samples are available on the samples page[2].

Further inspection of the `full-13049` lexicon reveals a potential reason why words containing a stop followed by /h/ are mispronounced without the use of syllable boundaries. The phoneme /h/ occurs mid-word in just 101 word types and only occurs after a stop in 10 of those. These word types have very low token frequencies in LJ Speech, ranging from 1 to 4, which add up to insufficient examples for the model to learn how to correctly pronounce the voiceless glottal fricative /h/ mid-word after a stop. Syllable boundary information resolves these cases presumably because they help the model learn to generalise from /h/ following non-stops to /h/ following stops.

`stress` did not lead to improved performance. It is worse (**In-LJ**:31/50, **Out-LJ**:33/50) than `full-13049`. `stress-syllable` (**In-LJ**:46/50, **Out-LJ**:45/50) is also inferior to `syllable`. In contrast to syllable information, lexical stress information is not beneficial to pronunciation control in our results.

Most models correctly pronounced 47-50 of the words in the **Phonemicised** test set (Section 5.2.4). That is, representation mixing doesn't negatively impact pronunciations that a `grapheme-only` model already pronounces correctly.

Another potential application of representation mixing models is to use phonemes

[2]Samples available at jonojace.github.io/IS20-repmixing-limits

to control pronunciations *within* the grapheme representation of a word, making pronunciation correction perhaps more accessible to non-expert users. For example, controlling the pronunciation of the first vowel in the homograph 'gala' by inputting 'g/aa/la' (UK) or 'g/ei/la' (US), , or correcting the mispronunciation at the morpheme boundary in 'loophole' by inputting 'loo/p | h/ole'. We generated various such samples from `syllable`, and informally confirmed that this type of correction is indeed possible despite graphemes and phones never being mixed *within* words during training.

5.4 Conclusion

In this chapter, we showed that representation mixing models cannot correct pronunciations with 100% accuracy by using phoneme inputs, even when the majority of word types are phonemicised during training by a large pronunciation lexicon. Despite this, they still control pronunciations relatively well, and we demonstrate via an exploration of principled word type selection methods that performance remains competitive even when using resource-limited lexica as small as just 500 entries. This makes representation mixing a cost-effective paradigm for correcting mispronunciations made by E2E TTS systems, which is of the upmost importance in low-resource scenarios.

Our experiments showed that when developing a lexicon for a representation mixing TTS system, choosing to phonemically transcribe word types based solely on the *number* of tokens they cover in the speech corpus is not optimal for pronunciation correction. Instead the *choice* of word types is also important. Our results indicate that choosing words with rich graphemic context helps greatly, possibly as rich graphemic context is a proxy for rich phonemic context.

In the next chapter, we move away from controlling TTS pronunciations using phoneme sequences, because transcribing phoneme sequences would be unfeasible in some languages. Instead, we move towards solutions that do not require phoneme sequences at training or inference time to control pronunciation. To achieve this, we begin our investigations into using recorded speech examples to represent and thus control the pronunciations of individual words. In the next chapter we use a neural vocoder to generate speech waveforms from self-supervised speech representations. We explore whether time-aligned sequences of speech

representations can be rearranged, and then vocoded to generate intelligible speech.

Chapter 6

Temporal sensitivity of discrete speech representations

This chapter represents an extended version of my research presented in *Analysing Temporal Sensitivity of VQ-VAE Sub-Phone Codebooks*, which was published in the proceedings of the Speech Synthesis Workshop 2021 (Fong et al., 2021). This was joint work done in collaboration with Jennifer Williams and Simon King.

6.1 Introduction

In the previous chapter, we proposed and demonstrated a method of pronunciation control that works well with small phoneme-based lexica. This novel paradigm, enabled by representation mixing of graphemes and phonemes, is very promising for under-resourced text-to-speech (TTS) scenarios. However, one downside of this approach is that it nevertheless requires *some* phoneme transcriptions. This renders it infeasible for languages for which no linguists are available.

Acknowledging this downside, in this chapter, we begin our explorations into using *speech* to control pronunciation. The upside of this approach is that linguists are *not* required. Instead, word pronunciations can be extracted from existing speech corpora, or can be recorded by any native speaker of a language. However, speech-based approaches to pronunciation control have, until now, been rather limited. One such approach is to concatenate recordings of whole words together to form the output utterance (Hunt and Black, 1996). Such word-based concatenation

can happen in unit selection systems when the desired output wordtype, and its linguistic context, and its predicted acoustic features match an occurrence of the same wordtype within the training data.

There is, however, a major downside to this audio concatenation approach. Speech contains many different channels of information aside from linguistic content, such as speaker, recording conditions, and emotion. The recorded speech must also match along *all* of these channels in order to blend in seamlessly and produce natural-sounding speech. In this chapter, we explore a novel approach where pronunciation corrections could come from a wider variety of sources, perhaps being spoken by a different speaker or being recorded in different conditions. We achieve this by using discretised speech representations rather than waveforms, to represent pronunciation.

Speech applications such as automatic speech recognition (ASR) and TTS have traditionally employed phonemes to describe speech. While phonemes were originally conceived for rapid linguistic field transcription, they are now widely used as a representation of pronunciation. Phonemes omit much of the nuance that is inherent in human speech production. For example, phonemes do not represent the effects of co-articulation and are inadequate for capturing other connected speech effects such as vowel reduction (i.e. the "**u**" in "ed**u**cation" moving towards the schwa [ə]) or elision (e.g. "I am" becoming "I'm" in english, and "Je aime" becoming "J'aime" in French). In speech applications, there are established approaches to work around the limitations of phonemes, including expanding the phoneme set to include allophones and syllabic consonants, or by constructing a set of application-specific context-dependent categories such as diphones (for TTS), or triphones / quinphones (for ASR). Until more recently, these were the only viable choices for a discrete representation of speech in speech technology applications.

However, recent advances in neural modelling, notably self-supervised and semi-supervised techniques, offer an ability to learn speech representations which – by definition – *must* capture nuances of speech since the training objective is to be able to reconstruct speech (Hsu et al., 2017; van den Oord et al., 2017) or make contextual predictions (van den Oord et al., 2018; Chung and Glass, 2020; Baevski et al., 2020). Since 2017, the prior body of work had already shown that speech representation models produce features that correlate with linguistic aspects of

speech (such as phones) from just speech waveforms. These representations greatly improve ASR with little data (Baevski et al., 2020), and can also be used to generate speech (Hayashi and Watanabe, 2020; Polyak et al., 2021). But no work had sought to use these representations to create new functionalities in existing speech applications.

We sought to explore how such informationally-dense speech representations could be used for TTS pronunciation control. In this chapter, we investigate whether sequences of speech representations can be spliced (at word boundaries), rearranged, and then vocoded without heavily impacting naturalness and intelligibility. More specifically, we encode speech into a sequence of tokens drawn from a finite set of tokens using a speech VQ-VAE model (van den Oord et al., 2017). Then, to evaluate the adequacy of the token sequence as a representation of speech pronunciation we concatenated token sequences to construct word sequences not seen in the training data. That is, we used "concatenative VQ-VAE synthesis" as a methodology for evaluating whether the learned inventory of tokens would be a useful pronunciation representation for neural TTS.

The main contribution of this chapter is measuring the extent to which VQ-VAE token sequences can be manipulated. It is desirable that the tokens correspond to speech in a monotonic and predictable manner. However, because the tokens are always learned as a temporal sequence from natural speech, they potentially have temporal sensitivity: they are not guaranteed to have a monotonic relationship to the speech signal, and they might be context- or even speaker-dependent. We offer what we believe to be the first demonstration that it is feasible to manipulate and synthesise from token sequences. We analyse increasingly challenging tasks, from copy synthesis to the production of novel speech by concatenating short phrases with matched vs. mismatched phonemic context and speaker identity.

6.2 Related work

6.2.1 Coarseness of input features

GORDOS et al. (1992) presents a method for developing a text-to-speech (TTS) system specifically for Modern Standard Arabic (MSA) using formant synthesis. The study identifies and defines a set of acoustic building units that capture

the essential phonetic characteristics of MSA. These units are used to synthesise speech by combining them in various ways to produce natural-sounding output.

Perquin et al. (2020) investigated whether using graphemes or phonemes affected the learned pronunciations in Tacotron 2 sequence-to-sequence TTS. They found that the internal representations for graphemes and phonemes were consistent, suggesting it is possible to control pronunciation directly from graphemes. However, when compared to frame-rate based self-supervised speech representations, graphemes are a relatively large unit and the corresponding neural embeddings may not be able to control nuances of pronunciation. In our work, we modelled a smaller unit with VQ-VAE tokens which could provide much finer control over pronunciation over grapheme embeddings.

6.2.2 The usage of self-supervised speech representations in speech applications

A growing area of interest involves methods to discover meaningful acoustic units from speech, often in an self-supervised or semi-supervised manner, and then utilise them in downstream tasks. Tjandra et al. (2020) explored Transformer VQ-VAE for zero-shot synthesis: generating speech from novel speakers without text or phone labels. They showed that the VQ-VAE architecture is able to discover sub-phone units that can be synthesised into high-quality speech. While this work is encouraging, they did not manipulate the sub-phone units directly, which is something that we explore in this chapter.

At the time of publication, circa 2021, one of the best examples of how VQ-VAE tokens could be applied to speech applications comes from the text-to-speech system called DiscreTalk (Hayashi and Watanabe, 2020). There, they experimented with generating VQ-VAE tokens with different down-sampling factors, effectively controlling the duration and frame rate of the VQ-VAE tokens. They trained a neural machine translation model to predict a sequence of VQ-VAE tokens from text input (similar to grapheme-to-phoneme prediction in conventional TTS systems). The predicted VQ-VAE tokens were then consumed by the VQ-VAE decoder to produce speech. They showed that tokens of longer duration improved training stability but at the cost of sacrificing overall speech quality. While this finding is important, it is not clear how the duration of the token affects more

nuanced elements of pronunciation such as co-articulation. Furthermore, the process by which the VQ-VAE tokens are predicted from text by their neural machine translation model is uncontrollable, resulting in a lack of control over pronunciations. In this chapter we make the crucial first steps towards ascertaining whether VQ-VAE tokens specifically, and neural speech representations more generally, are a good candidate for controlling the pronunciation of synthesised speech.

6.3 Data and model

6.3.1 Data

To explore whether multispeaker pronunciation corrections would be possible we used the multispeaker VCTK dataset (Yamagishi et al., 2019) to both train our VQ-VAE model and generate samples for our listening test. Although it is a relatively small dataset (approximately 44 hours over 109 speakers), since the recordings are of high quality, and our WaveRNN decoder (Kalchbrenner et al., 2018) is of sufficient model capacity, our resulting system is able to accurately generate speech for each of the VCTK speakers. VCTK contains voices from a variety of speakers of different ages and UK accents.

While obtaining 44 hours of *studio quality* speech from over 109 speakers may be feasible in low resource scenarios, we believe that this is feasible if crowd sourcing methods are used to collect speech data.

6.3.2 Discrete self-supervised learning speech representation model

To discover discrete tokens into which speech can be encoded, and then subsequently synthesised, we used a VQ-VAE model (van den Oord et al., 2017). Our model differed from the original by using WaveRNN[1] as the vocoder instead of WaveNet (van den Oord et al., 2016), for faster training and inference.

The VQ-VAE encoder used 10 one-dimensional convolutional layers to encode a sequence of waveform samples $\mathbf{x}_{1:T}$ into a sequence of 128-dim continuous latents $\mathbf{z}_{1:S}$ ('T' are waveform sample timesteps occurring at the audio sampling rate, and

[1] https://github.com/mkotha/WaveRNN

'S' are latent timesteps occurring at a lower frequency according to the temporal down-sampling performed by the encoder). These latents were then discretised using a vector quantisation layer to create a sequence of code-words (i.e., codebook entries) $\mathbf{d}_{1:S}$, which henceforth we will call **tokens**. Our codebook contained 512 128-dim entries. A WaveRNN decoder produced waveform samples at 22.05 kHz sample rate, conditioned on the sequence of tokens and $\mathbf{s}_{1:S}$, a single speaker one-hot vector that is broadcast across all timesteps.

Since the focus of this study was to explore concatenative synthesis and *not* to examine VQ-VAE's ability to generalise to unseen speakers, we chose to train the model on all of VCTK's speakers. Thus we performed concatenative VQ-VAE synthesis from parts of training utterances, just as would be the case in unit selection (Hunt and Black, 1996). We trained our model using all VCTK speakers for 1000 epochs (2.7 million iterations), which took approximately 1 week on a single NVIDIA 2080Ti GPU.

6.4 Method

In order to find a way to meaningfully manipulate speech in the VQ-VAE token domain, we choose work at the word-level because it is a first approximation to being able to manipulate VQ-VAE tokens at a lower level (such as below the phone-level). A successful outcome at the word-level would signify that VQ-VAE tokens may be a feasible representation for controlling pronunciation. Furthermore we re-arranged VQ-VAE token sequences that correspond to 3-word chunks, since we believed that longer sequences of conditioning VQ-VAE tokens would be more likely to generate output from the vocoder.

Figure 6.1 shows an overview of the concatenative VQ-VAE synthesis pipeline that extracts tokens corresponding to words in two separate source utterances, these tokens are then used to generate a single target utterance stimulus. The extraction process for one source utterance works as follows: we first encode the waveform of utterance U_1 into a sequence of tokens $\mathbf{d}_{1:S}$ using the VQ-VAE encoder. We then use the timestamps for a sub-sequence of words (e.g. w_1, w_2, w_3) to identify and extract their corresponding VQ-VAE tokens $\mathbf{d}_{left}$. We then repeat this process for a second source utterance U_2 to get another sequence of tokens $\mathbf{d}_{right}$ corresponding to w_4, w_5, w_6. We then concatenate them together to get

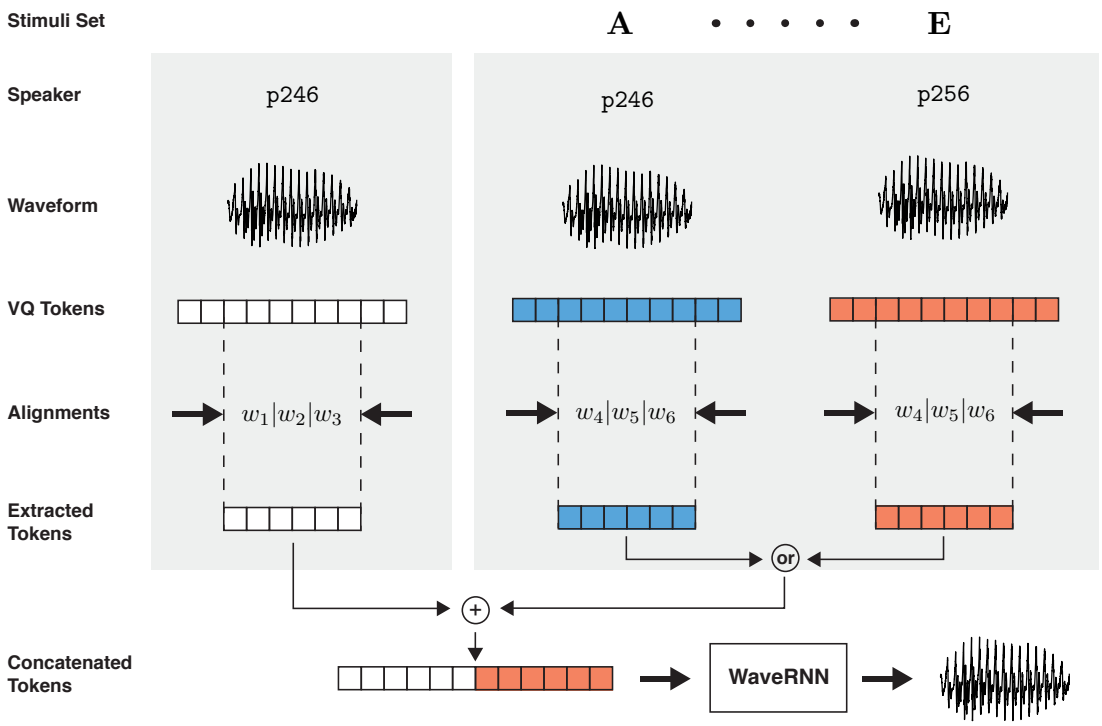

Figure 6.1: Overview of the concatenative VQ-VAE synthesis method. The token sequence corresponding to word sequence w_1, w_2, w_3 is being concatenated with the token sequence corresponding to w_4, w_5, w_6, thus creating the utterance $w_1, w_2, w_3, w_4, w_5, w_6$, from which a waveform is synthesised using WaveRNN. Depicted in the diagram is a set $\mathbf{A}$ stimulus being generated for speaker p246 and a $\mathbf{E}$ stimulus being generated for the p246+p256 speaker combination. The particular sub-sequence of words w_4, w_5, w_6 used depends on the stimulus set currently being generated, either $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$, or $\mathbf{E}$. These stimulus sets are detailed in Section 6.5.2. Note, the four waveforms depicted are identical, but in reality they will all be different.

$\mathbf{d}_{left} \bigoplus \mathbf{d}_{right}$, and use that sequence to condition the WaveRNN to generate speech for $w_1, w_2, w_3 \bigoplus w_4, w_5, w_6$. '$\bigoplus$' signifies concatenation of sequences of words or tokens.

Word-level alignments are required to find the correspondence between tokens and words, and are found using the Montreal forced aligner (McAuliffe et al., 2017).

We also found that the generation of words corresponding to the start of $\mathbf{d}_{left}$ and the end of $\mathbf{d}_{right}$ were sometimes cut-off or not realised, This is likely due to the word sequences being extracted from connected speech. Subsequently we experimented with inserting $\mathbf{d}_{sil}$ before $\mathbf{d}_{left}$ and after $\mathbf{d}_{right}$, finding that it helps recover some words. $\mathbf{d}_{sil}$ is a 128-dim vector of zeros. When generating our listening test stimuli we pad by 50 tokens of $\mathbf{d}_{sil}$ on both sides. We also chose to use 6-word stimuli rather than 4-word ones so that the words adjacent to those at the concatenation point are not cut off in any way and should be synthesised completely (i.e. in $w_1, w_2, w_3 \bigoplus w_4, w_5, w_6$, w_2 is flanked by w_1 and w_3, and w_5 is flanked by w_4 and w_6. Generated speech can be found on our samples page[2].

6.5 Experiments

6.5.1 Word transcription task

We designed our experiment to answer the following three research questions:

- RQ1: Does concatenative synthesis result in less intelligible speech than copy-synthesis (Set **A** vs. Sets **B**, **C**, **D**, **E** in Section 6.5.2)?
- RQ2: Is intelligibility affected by extracting tokens from speech spoken by different speakers (Sets **B**, **D** vs. Sets **C**, **E** in Section 6.5.2)?
- RQ3: Is intelligibility affected when the two words adjacent to the concatenation boundary, w_3 and w_4, are chosen so that their linguistic contexts, according to their surrounding words in their original sentences, mismatch (Sets **B**, **C** vs. Sets **D**, **E** in Section 6.5.2)?

In order to test whether re-arranging and synthesising sequences yielded intelligible speech, especially at the concatenation boundary we used a fill-in-the-blank transcription task. We generated 6-word stimuli of the form $w_1, w_2, w_3 \bigoplus w_4, w_5, w_6$.

[2] https://jonojace.github.io/SSW21-concatenative-vqvae

For each stimulus, participants were asked to transcribe the word *after* the concatenation point (i.e., w_4) by being presented with the transcription *minus* the word-in-question. For example, for the utterance 'red and green $\bigoplus$ looking any further' using $\mathbf{d}_{left}$ corresponding to 'red and green' and $\mathbf{d}_{right}$ to 'looking any further', participants were presented with the transcription 'red and green `<blank>` any further'.

6.5.2 Experimental conditions

Our listening test contained 5 stimulus sets each with 40 stimuli. Each participant rated the same 200 stimuli. We determine whether linguistic context matches between the two words at a concatenation boundary by comparing their two triphones at the boundary. For example if w_3 is 'hello' (HH EH L OW) and w_4 is 'world' (W ER L D), then we compare the rightmost triphone of 'hello' to the leftmost one of 'world'. If 'hello' and 'world' occurred together in the recording, then the rightmost triphone of 'hello' is L+OW L-OW+W **OW-W**, and the leftmost triphone of 'world' is **OW+W** OW-W+ER W-ER and we would consider them having a matching linguistic context. If instead 'world' was preceded by a word ending in AH in the recording, then its leftmost triphone would be AH+W AH-W+ER W-ER and we would consider its linguistic context to mismatch with the recording of 'hello' previously mentioned. The following is a description of the characteristics of each stimulus set:

- **A**: Copy-synthesis. The speech corresponding to a contiguous sequence of words from a single source utterance, $w_1, w_2, w_3, w_4, w_5, w_6$, is encoded by the model, and then decoded to a waveform.
- **B**: Matched context + Matched Speaker. The speech corresponding to two sequences of words from two different source utterances, w_1, w_2, w_3 and w_4, w_5, w_6 is extracted, encoded by the model, concatenated together to form $\mathbf{d}_{left} \bigoplus \mathbf{d}_{right}$, and then decoded to a waveform. The speaker of the two utterances is identical, and the linguistic context of the two words at the concatenation point match.
- **C**: Matched context + Mismatched Speaker. Similar to **B**, but the speakers of the two utterances differ.
- **D**: Mismatched context + Matched Speaker. Similar to **B**, but the linguistic

context of the two words at the concatenation point mismatch.

- **E**: Mismatched context + Mismatched Speaker. Similar to **B**, but the speakers of the two utterances differ, and the linguistic context of the two words at the concatenation point mismatch.

6.5.3 Stimuli

We used 40 unique sequences w_1, w_2, w_3 each of which could be followed by one of 5 unique sequences w_4, w_5, w_6 (thereby creating 200 unique utterances in total, described in 6.5.2). Each unique sequence w_1, w_2, w_3 was presented 5 times during the listening test (once per stimulus set), and each time is coupled with a unique w_4, w_5, w_6, making a total of 200 utterances, noting that these may not all be grammatically-correct. When choosing a w_4, w_5, w_6 for a given w_1, w_2, w_3 we make sure not to choose those which contain a w_4 that is either a stopword or a word that has been generated before for a particular speaker.

6.5.4 Speakers

We took care in choosing the speakers to build our stimuli. We found in preliminary experiments that conditioning the WaveRNN using tokens extracted from slower-speaking voices resulted in more intelligible speech, when performing either copy-synthesis or concatenative reconstructions. Subsequently we chose 4 slow-speaking voices for our experiments; `p246`, `p256`, `p345`, and `p374` whose speaking rates are summarised in Table 6.1.

Each of the 4 speakers was used to generate a subset of 50 sentences from the set of 200 unique sentences (with no overlap between each speaker). The 50 stimuli for a given speaker is made up of 10 stimuli from each stimulus set (**A**, **B**, **C**, **D**, and **E**). Here is an example of how the 50 stimuli for a single main speaker is generated: supposing that the main speaker is `p246` and the secondary speaker is `p256` (i.e. for answering RQ2 regarding mismatched speakers) then we will generate 10 stimuli for each set **A**, **B**, and **D** using tokens only from `p246`, and 10 stimuli for each set **C** and **E** using $\mathbf{d}_{left}$ from `p246` and $\mathbf{d}_{right}$ from `p256`. The 4 main and secondary speaker combinations that we use are `p246+p256`, `p256+p345`, `p345+p374`, and `p374+p246`. To condition the WaveRNN we always use the one-hot vector corresponding to the main speaker to condition all of the

Table 6.1: Speaking rate information (average number of seconds per phone).

Duration Type	p246	p256	p345	p374
non-sil Phone	0.088	0.085	0.083	0.118
sil Phone	0.114	0.106	0.278	0.278

tokens, $\mathbf{d}_{left} \bigoplus \mathbf{d}_{right}$, even if the original speaker of $\mathbf{d}_{right}$ is the secondary speaker. Therefore, our model performs voice conversion from the secondary speaker to the main speaker on the latter halves of the tokens, $\mathbf{d}_{right}$, in sets **C** and **E**.

6.5.5 Listening test

We built our listening test[3] on the Qualtrics platform[4], and recruited participants using Prolific[5]. We recruited 50 participants, each of whom were from the UK, have no literacy difficulties, and have at least a 90% approval rating on Prolific. The order of the 200 stimuli were randomised on a per participant basis.

In order to ensure that our results were accurate we performed a manual check of all participants' answers to correct misspellings (e.g. 'contenders' and 'contendors'), typos (e.g. 'fresh' and 'frsh'), and homophone ambiguity errors (e.g. 'rode', 'rowed', and 'road').

6.6 Results

6.6.1 stimulus set comparisons

We present results from our intelligibility test partitioned across each stimulus set in Figure 6.2. We found that the copy-synthesis stimuli (**A**) were the most intelligible. This was likely because no concatenative synthesis is performed, and as such the resulting token sequences will not suffer from potential alignment errors and will consequently match the training conditions of the WaveRNN, which was only trained on natural sequences of VQ-VAE tokens and their corresponding waveforms.

[3]Test building automated using https://github.com/CSTR-Edinburgh/qualtreats
[4]https://www.qualtrics.com/uk/
[5]https://www.prolific.co/

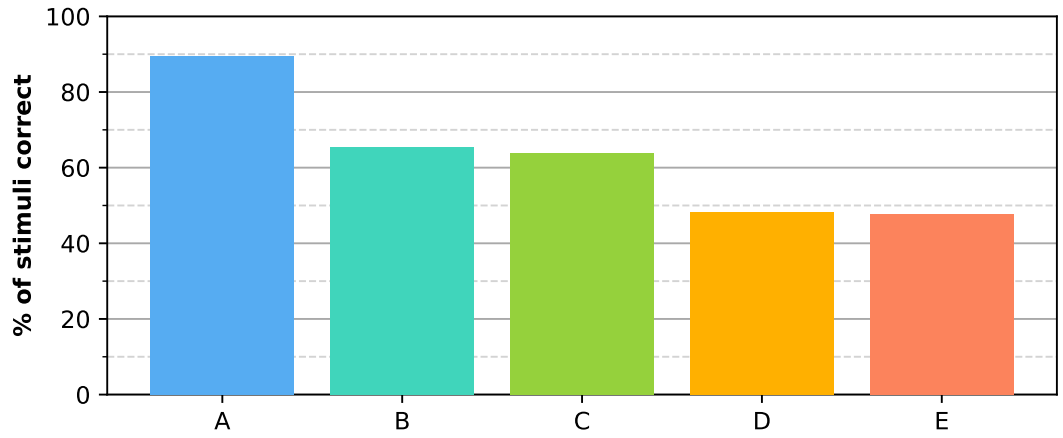

Figure 6.2: Intelligibility results across the stimulus sets **A**, **B**, **C**, **D**, **E** described in Section 6.5.2. We present the percentage of words within a set identified correctly by participants.

The results of **B** and **C** show that concatenative VQ-VAE synthesis can produce intelligible speech somewhat reliably. Additionally since **B** and **C** outperform **D** and **E** we can conclude that concatenative synthesis works better when linguistic contexts match at the concatenation boundary.

Comparing the results of **B** vs **C** and **D** vs **E** we find that synthesising using concatenated token sub-sequences extracted from different speakers has only a very small negative impact on intelligibility. The closeness of these results combined with our observations that the speaker identity of generated samples does not audibly change mid sentence is testament to the ability of VQ-VAE to learn codebook embeddings that are disentangled from speaker identity (even with a relatively large codebook size of 512 entries). Disentanglement between speaker and linguistic content is achieved due to the use of explicit ground-truth speaker conditioning to the decoder and the extreme bottle-necking of the input signal in both the time and feature dimensions (resulting in a low bit-rate encoding). These results are promising for future concatenative neural synthesis work, demonstrating that it is possible to mix and match VQ-VAE tokens between different speakers. This enables correcting a system's pronunciations via cheap-to-obtain speech exemplars, obtained from a different speaker, rather than more expensive phonemic transcriptions.

6.6.2 Qualitative analysis

In this section we present our findings regarding the *types* of errors that participants made. We were particularly interested in the cases of incorrect transcription because it helps us better understand how well we might be able to manipulate VQ-VAE tokens at a finer granularity. One of our hypotheses was that concatenating $\mathbf{d}_{left}$ and $\mathbf{d}_{right}$ together would mainly cause the sounds adjacent to the boundary to be affected. Surprisingly however we found that there are instances where the first phone of w_4 seemed to be not correctly heard, but the rest of the word was largely intelligible, causing participants to hallucinate an incorrect answer. Since we did not discover a cause for this phenomena we include examples on our samples page and leave further investigation to future work.

6.7 Conclusion

In this chapter we presented an analysis of the sensitivity of VQ-VAE tokens to their surrounding context by using concatenations of tokens extracted from disparate sentences to generate speech. We found that 'unit selection' speech generation is possible in the discrete latent space of VQ-VAE tokens. Furthermore, by extracting VQ-VAE tokens from sentences selected from a variety of specific conditions we discover that VQ-VAE tokens are highly linguistic context dependent, but not very speaker context dependent. Together these two results are promising for future speech systems as they suggest that readily available audio exemplars can be used to alter aspects of speech such as pronunciation without resorting to expensive hand-transcribed labels such as phonemic transcriptions. We further observe that, within our pipeline, speaking rate and duration of silences can affect downstream reconstruction.

With regards to future work, there are several research directions which could either help understand or improve the method of concatenative VQ-VAE synthesis explored in this chapter. One could investigate neural concatenative synthesis cross-lingually, make tokens less context dependent without sacrificing reconstruction quality, and remove the reliance of the system on pretrained forced aligners and instead use word-level alignments obtained in an unsupervised fashion.

In the next chapter, we build upon the preliminary work done in this chapter to propose a more practical approach. We introduce a novel TTS model that can

use *either* graphemes or speech to represent pronunciation. The model will be trained on inputs composed of graphemes and speech, in a similar fashion to the grapheme-phoneme representation mixing of Chapter 4, so that speech exemplars can be used at synthesis time to correct pronunciation. Our results will show that this approach is advantageous at it greatly reduces the prevalence of audible join artifacts which were exhibited by the model investigated in the current chapter.

Chapter 7

Correcting text-to-speech mispronunciations using spoken examples

This chapter is an expanded version of a conference paper *Speech Audio Corrector: using speech from non-target speakers for one-off correction of mispronunciations in grapheme-input text-to-speech*, (Fong et al., 2022a) accepted at Interspeech 2022, Seoul, South Korea. This was joint work done in collaboration with Daniel Lyth, Gustav Eje Henter, Hao Tang, and Simon King.

7.1 Introduction

In this thesis, prior to this chapter, we have explored two approaches in an attempt to control TTS pronunciations while using as few linguistic resources as possible.

Firstly, in Chapter 5 we adopted the representation mixing paradigm where grapheme inputs are replaced with phoneme inputs when greater control over pronunciation is required. Within this paradigm we explored whether pronunciation control is retained when reducing the amount of phonemic resources that are available and found that reducing the amount of phoneme transcriptions available during training has an adverse impact on pronunciation control reliability. This reveals a cost-benefit trade-off: increasing phonemic resources increases control, but also increases cost.

Secondly, in Chapter 6 we directly conditioned a neural vocoder on spliced and concatenated sequences of speech representations at inference time. We found that doing so could result in poor sounding join artifacts, especially when the linguistic triphone context of spliced-in speech representations did not match their surroundings. This limitation was likely induced by the model's inability to utilise concatenated sequences of *contextual* representations that were stripped of their original context. In other words, this model was not trained to concatenate sequences of speech representations together and reconstruct them in a smooth and coherent manner.

The approaches in Chapter 5 and Chapter 6 exhibit significant limitations, hence they are not *complete* solutions to pronunciation control in low-resource settings.

In this chapter, we introduce our research inspired by the findings of the previous two chapters. We introduce a novel approach, named Speech Audio Corrector (SAC), that directly uses speech during training and inference, thereby forgoing the need for phonemes. The approach can make one-off corrections of word mispronunciations at inference time without requiring *any* model retraining or fine-tuning. The approach is enabled by tackling the train and inference time disparity observed in Chapter 6 by training the TTS model to accept input sequences of graphemes that may or may not also include speech codes. To induce this capability, we substitute words' graphemes for their speech codes randomly during training. In doing so, we observe that this trains the model to generate smooth transitions in the predicted mel-spectrograms, even though its inputs may involve any combination of grapheme and speech code sequences.

Due to the added complexity and data requirements of modelling speech from scratch, our SAC model represents pronunciations as word-aligned *speech codes* instead of raw speech waveforms or engineered features such as MFCCs. Speech codes are discretised representations extracted from self-supervised learning models pretrained on large amounts of speech data. Speech codes have potential for controlling pronunciation as they have been shown to correlate well with phonemes (van den Oord et al., 2017; Baevski et al., 2020; Hsu et al., 2021) and yet desirably are relatively free of speaker information (Kharitonov et al., 2022). Furthermore, speech codes can be considered as relatively low-resource representations because self-supervised learning models can be trained on speech data only, without requiring text or phoneme transcriptions.

With regards to impact, our speech-based approach to pronunciation control has the potential to lower the barrier to entry for building TTS voices by making it possible to use a pronunciation lexicon constructed from spoken examples extracted from existing corpora, or by crowd-sourcing from untrained speakers. This approach would be feasible for any language in the world, and would help make voice building become more accessible to local communities. In contrast, constructing phoneme-based lexicons is a more expensive endeavour as doing so requires highly-trained linguists and speech engineers, who do not exist for every language.

In this chapter: We demonstrate that SAC can correct a word's pronunciation using speech obtained from a non-target speaker regardless of their accent. We also find that matching the speech correction's accent to the target voice is preferred to using a mismatching accent. Finally, we find that adding SAC's corrective functionality has little impact on standard grapheme-input TTS pronunciations.

7.2 Background

7.2.1 Sequence-to-sequence transformer-based text-to-speech

As discussed in Section 2.5.2.2, the Transformer has emerged as a prominent neural network architecture for almost all natural language processing (NLP) tasks, such as self-supervised representation learning, machine translation, and text generation. The Transformer's remarkable performance can be attributed in part to its proficiency in capturing long-range dependencies when producing contextual representations and its computational efficiency during both training and inference, owing to forward pass parallelisation. Consequently, the Transformer has largely supplanted Recurrent Neural Networks (RNNs) in tasks that involve generating contextual representations from lengthy input sequences.

Moreover, Transformers have been applied successfully to Text-to-Speech (TTS) tasks. The initial approach introduced by Li et al. (2019), known as Transformer TTS, employs a Transformer encoder equipped with self-attention to encode textual or linguistic feature inputs into contextual representations. Subsequently, these representations are decoded into mel-spectrograms using an autoregressive Transformer decoder. This architectural framework bears resemblance to that

of Tacotron 2 introduced by Shen et al. (2018b). According to Li et al. (2019) Transformer TTS enhances TTS synthesis by effectively capturing long-range dependencies at the prosodic level, while also offering notable computational advantages during training and inference processes.

For our specific objectives, where the utilisation of speech inputs to control the pronunciation of mispronounced words is desired, the self-attention mechanism of the Transformer encoder is critically important. By enabling the generation of contextual representations without limitations associated with temporal proximity (a property that CNNs and RNNs exhibit), the self-attention mechanism allows for flexible exploration of various methods to incorporate speech into the textual input, such as appending it after the text and/or utilising masking tokens. This input approach, where text comes first and speech comes second, is illustrated in Figure 7.1.

7.2.2 Generation from self-supervised speech representations

Speech representations obtained through self-supervised learning (SSL) techniques, including models like HuBERT (Hsu et al., 2021), wav2vec2.0 (Baevski et al., 2020), and VQ-VAE (van den Oord et al., 2017), have been extensively utilised for various downstream tasks. These tasks encompass speech recognition (Hsu et al., 2021; Baevski et al., 2020), speech coding (Dunbar et al., 2019; Polyak et al., 2021), voice conversion (van den Oord et al., 2017; Polyak et al., 2021), natural language generation (Lakhotia et al., 2021; Kharitonov et al., 2021, 2022), and speech-to-speech translation (Lee et al., 2021a,b). The widespread adoption of continuous SSL representations in such applications stems from their capability to model and extract high-level speech characteristics, such as phoneme category and speaker identity, in an unsupervised fashion. This capability is in stark contrast to traditional engineered features such as MFCCs. Consequently, SSL models can be fine-tuned to achieve strong performance in diverse tasks such as Automatic Speech Recognition (ASR), Keyword Spotting, and even semantic intent classification or slot filling (Yang et al., 2021).

Nevertheless, despite their widespread adoption in various speech applications, SSL representations have not been utilised to enhance the *pronunciation* capabilities

of neural Text-to-Speech (TTS) systems. While SSL representations have been employed as intermediate features in TTS, their usage has primarily been driven by the objective of reducing reliance on engineered features (Hayashi and Watanabe, 2020), improving duration modelling (Yasuda et al., 2021a), and introducing increased prosodic variation (Henter et al., 2018; Sun et al., 2020). To the best of our knowledge, the work presented in this chapter represents the first instance of leveraging the phonetic information embedded within discrete speech codes to exert control over pronunciations in grapheme-input TTS systems.

7.2.3 Text-to-speech pronunciation control via speech recordings

Unit selection, a well-established paradigm in Text-to-Speech (TTS), involves the pronunciation of words by selecting and concatenating recorded speech fragments, known as 'units', from a database. These units are chosen based on their labels matching the corresponding text to be spoken (Hunt and Black, 1996). It is worth noting that the labels serve solely as indices into the database and do not necessarily represent fine acoustic details or prosody. These aspects are naturally obtained by selecting units within an appropriate context. However, when concatenating units on the output side, undesirable join artifacts arise, which cannot always be resolved through signal processing techniques. Furthermore, the unit database must consist of speech from a single speaker, making it infeasible to incorporate or add non-target speaker data at a later stage.

In contrast, our proposed model, Speech Audio Corrector, predicts mel-spectrograms by utilising both linguistic and speech features. These features are concatenated on the input side, enabling SAC to synthesise speech with fewer join artifacts. Notably, SAC is specifically designed to facilitate pronunciation control using speech codes from non-target speakers while preserving the identity of the target synthesised voice.

While previous studies have employed speech codes directly to condition a vocoder for purposes such as bit-efficient speech encoding or voice conversion, our prior work, as described previously in Chapter 6, demonstrated that splicing and concatenating sequences of speech codes which directly condition a vocoder can lead to join artifacts resembling those encountered in unit selection. In contrast,

we observe that our SAC model learns, *during* training, to transition smoothly between sequences of graphemes and speech codes.

7.2.4 Multi-modal input for text-to-speech pronunciation control

The way that pronunciation control is achieved in this chapter is inspired by the representation mixing paradigm, where graphemes and phonemes are interleaved during training to enable synthesis primarily from grapheme input while offering the option of pronunciation control using phonemes at inference time. For this purpose Kastner et al. (2019) used Recurrent Neural Networks (RNNs), and in our previous work, described in Chapter 5, we used Convolutional Neural Networks (CNNs).

In the case of SAC, a transformer encoder utilises self-attention and word position information to replace masked graphemes with speech codes. The way that SAC incorporates text and speech modalities on the input side drew inspiration from the work of Jia et al. (2021) who introduced a text encoder of their PnG BERT model which leverages self-attention to augment phoneme sequences with graphemes, resulting in synthesised speech with more natural prosody.

Regarding the topic of training data, our previous work, presented in Chapter 5 has explored the feasibility of grapheme-phoneme representation mixing in low-resource scenarios. We discovered that robust pronunciation control is compromised when employing a small pronunciation dictionary, which covers only a limited percentage of the word types encountered during training. In other words, there is a trade-off between the cost and benefit of representation mixing: Is the cost of phonemically transcribing additional word types justified by the pronunciation control that is gained? In contrast, SAC circumvents this trade-off by ensuring that speech codes are available for *all* word types observed during training, effectively establishing an implicit word type-to-speech code "dictionary" with 100% coverage. This characteristic allows SAC to learn a reliable back-off model for pronunciation control without encountering any cost-benefit trade-off.

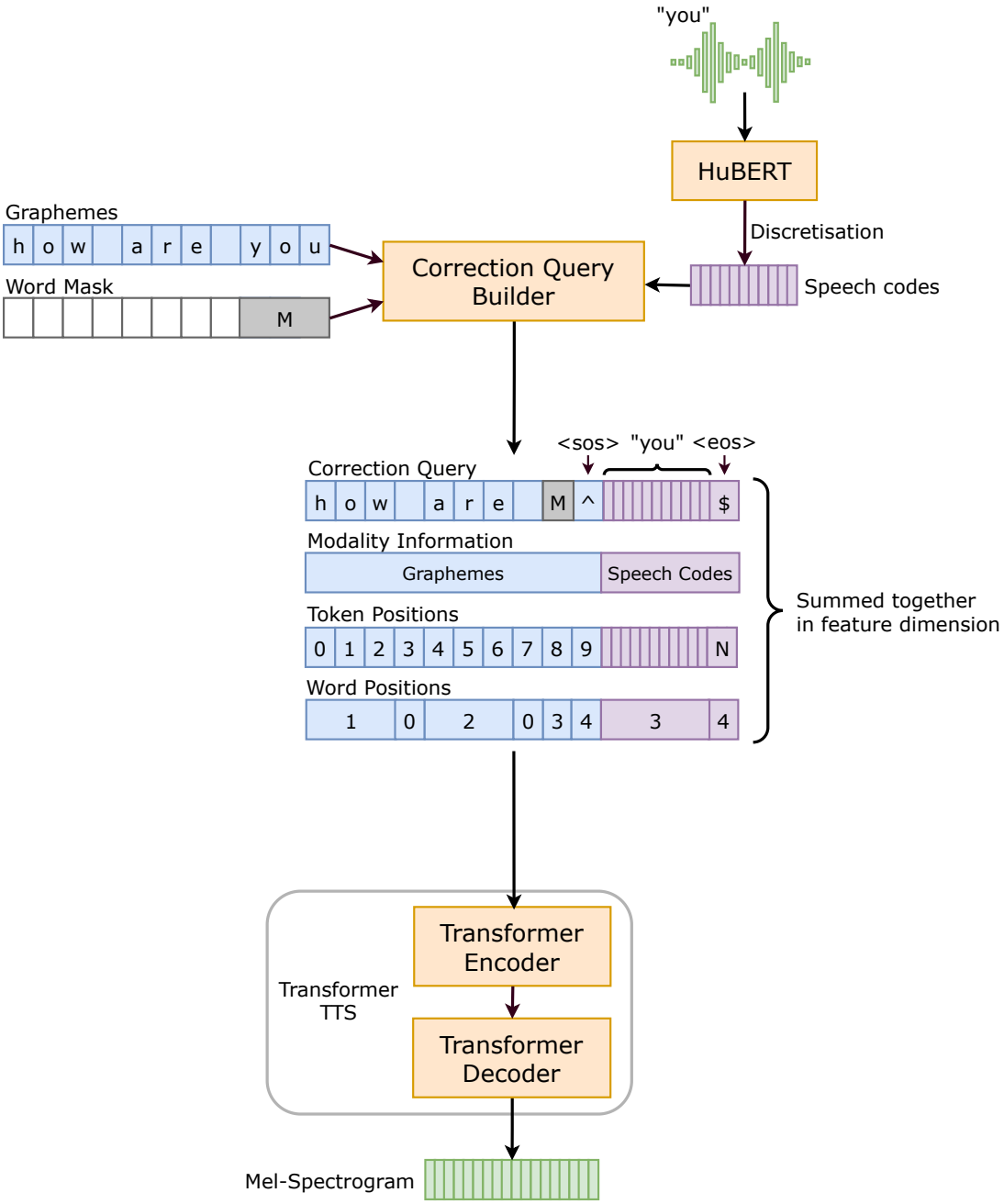

Figure 7.1: Architectural overview of Speech Audio Corrector (SAC) performing inference time one-off pronunciation correction of grapheme-input TTS using a recorded exemplar encoded as speech codes.

7.3 Our proposed model: Speech Audio Corrector

At the core of our proposed model Speech Audio Corrector (SAC), which is visualised in Figure 7.1, is Transformer TTS (Li et al., 2019), which is trained in an end-to-end fashion to predict mel-spectrograms from linguistic inputs – which are normally graphemes or phonemes. SAC adds the use of speech codes, which are discretised self-supervised learning speech representations, to enable a multi-modal correction query to control a word's pronunciation at inference time. A correction query, which is the input to SAC, comprises the grapheme sequence of the text to be synthesised, plus a sequence of speech codes for the word(s) whose pronunciation we wish to control, those words being masked out in the grapheme sequence. A correction query reduces to regular grapheme input when no speech codes are present. Additionally, auxiliary features are summed to the correction query. Token and word positional information is added to help the model align masked out graphemes to their speech codes. Finally, modality information is added to indicate to the model where grapheme and speech code sequences are within the correction query.

To form a correction query, a sequence of graphemes is processed by the Correction Query Builder module which in essence simply appends speech codes, and then sums auxilliary features. The module masks out the words that we wish to correct and retrieves their corresponding speech codes. The Correction Query Builder outputs a sequence of graphemes with mask tokens concatenated to a sequence of speech codes, forming $\mathbf{x_{1:N}}$ where $\mathbf{N}$ is the total number of timesteps of the correction query. The two sequences are each wrapped with start-of-sequence $\langle \text{sos} \rangle$ and end-of-sequence $\langle \text{eos} \rangle$ symbols. Hence, $\mathbf{N}$ is equal to the number of unmasked graphemes (including whitespace tokens), plus the number of masked words, plus the number of speech code timesteps, plus one for $\langle \text{sos} \rangle$, and plus one for $\langle \text{eos} \rangle$. Each of the symbols in $\mathbf{x_{1:N}}$ is looked up in a shared embedding table (shared between grapheme symbols and discretised speech codes), to obtain a sequence of embeddings $\mathbf{e_{1:N}}$.

Next, to the embedded correction query $\mathbf{e_{1:N}}$, we sum modality, token position, and word position features. Word position features help SAC align the masked-out graphemes with the corresponding speech codes. We embed token and word positions forming positional embeddings $\mathbf{tpe_{1:N}}$ and $\mathbf{wpe_{1:N}}$. Additionally the

word position embeddings are passed through a learned linear projection DENSE to distinguish them from the token positions. Modality embeddings $\mathbf{mode_{1:N}}$ are also employed to help the model differentiate where the grapheme and speech code sequences are located. Finally, learned weights α, β, and γ are applied to the auxiliary embeddings before summing them to $\mathbf{x_{1:N}}$. These allow the model to choose which features to prioritise. SAC's inputs are summarised in Equation 7.1. Once the inputs are formed and summed, the architecture and loss to be optimised during training are identical to Transformer TTS.

$$\mathbf{e_{1:N}} + \alpha \cdot \mathbf{tpe_{1:N}} + \beta \cdot \textsc{Dense}(\mathbf{wpe_{1:N}}) + \gamma \cdot \mathbf{mode_{1:N}} \tag{7.1}$$

7.4 Experimental setup

In this section, we detail choices made for the experiment regarding data, model training, model selection, and evaluation in order to test the following hypotheses:

- H_1: speech codes obtained from the speech of a *non-target speaker* can be used to make one-off pronunciation corrections
- H_2: speech codes from a non-target speaker with an accent *matching* that of the target speaker will provide more robust pronunciation corrections than for a *differing* accent

In H_2 we use the term *robust* pronunciation control. In this context we define robust to mean the degree to which intended pronunciations are realised by a system. For example a system that corrects pronunciations 100% of the time is deemed to be more robust than a system which successfully corrects pronunciations only half of the time.

The combined evaluation of H_1 and H_2 is pivotal in determining the feasibility of utilising SSLs to correct TTS pronunciations in practical applications. In real-world scenarios, it is not always possible to have access to the original voice talent who recorded the high-quality TTS training dataset. Consequently, an alternative speaker, potentially with a different accent, may need to be employed. By evaluating both H_1 and H_2, we gain valuable insights into the effectiveness and generalisability of using SSL representations to control TTS pronunciations across different speaker and accent conditions.

7.4.1 Data

To train SAC, we use text and speech from chapters 4 to 50 of the LJ Speech corpus (Ito and Johnson, 2017). We use chapter 3 of LJ Speech for our development set. LJ Speech contains 24 hours of speech from a single female US speaker. We use the standard normalised text transcription of LJ Speech.

In order to obtain word aligned speech codes from the recorded speech, we first extract 50 Hz layer-6 representations using HuBERT-Base-LS960[1] from 16 kHz speech. We use layer-6 representations as they have been found to correlate well with phone identities (Hsu et al., 2021) as discussed in Section 7.2.2. Next we discretise these representations into speech codes using a 100 clusters k-means model[2] which was trained on HuBERT-Base-LS960 representations extracted from librispeech 960h (Panayotov et al., 2015). Kharitonov et al. (2022) in their work demonstrated that the use of 100 clusters provides an appropriate trade-off between phonetic quality (measured by WER) and speaker-agnosticism (measured by speaker probing accuracy). Note that in this chapter, we used 100 clusters instead of the 512 clusters discussed in Section 6.3.2. This decision was made to reduce the number of clusters, thereby biasing the representations to exclude speaker-specific information and facilitate corrections from non-target speakers.

In this work, we chose to not remove duplicate speech codes from the resulting sequences. De-duplicating speech code sequences has been a common choice in recent generative spoken language modelling work (Lakhotia et al., 2021; Kharitonov et al., 2021) as doing so reduces computational load and increases modelling performance. However, we found that de-duplication resulted in unreliable pronunciation control when synthesising using speech codes, which may be due to code de-duplication placing extra burden on the attention-based duration modelling task implicit within Transformer TTS. Finally, we align speech codes at the word-level using Montreal Forced Aligner (McAuliffe et al., 2017). From this, we derive a dictionary comprising one-to-many mappings between word types and corresponding speech code sequences.

To test H_1 and H_2 we utilised the multi-speaker, multi-accent VCTK corpus (Veaux et al., 2017). We obtained word aligned speech codes from VCTK in an identical fashion to LJ Speech, with each word type typically being spoken by

[1] https://dl.fbaipublicfiles.com/hubert/hubert_base_ls960.pt
[2] https://dl.fbaipublicfiles.com/textless_nlp/gslm/hubert/km100/km.bin

multiple speakers across a variety of accents.

7.4.2 Model

We trained 2 models in total: a baseline Transformer TTS model (hereafter: TTS), and our proposed model (SAC). We used the Transformer TTS implementation by Ott et al. (2019) in fairseq, and implemented SAC on top of this. Consequently the core model architecture is identical between TTS and SAC and followed that of Transformer TTS introduced by Li et al. (2019): The transformer encoder and decoder both have 6 layers each with 4 attention heads, and a hidden size of 512 between all layers. The embeddings of the correction query, modality information, token position, and word position features each have 512 dimensions. Both TTS and SAC were trained without the attention guide loss introduced by Tachibana et al. (2018a) since there is no strict monotonic alignment between the correction query and output acoustics when speech codes are included within the query.

Both models were trained in an identical manner, with identical hyper-parameters such as learning rate and batch size. Both models use graphemes rather than phonemes in order to reflect a low-resource scenario. The training data for SAC comprised of randomly-constructed correction queries. For each word in each input grapheme sequence, we masked it out and provided the corresponding speech codes, with a 50% probability. In preliminary training runs we searched over hyper-parameters to find a SAC model that minimised Mel-Cepstrum Distortion (MCD) calculated over the LJ Speech development set when all words are represented by their respective speech codes rather than graphemes. Listening test stimuli were generated from epoch 1000 of TTS and SAC which minimised SAC's development set MCD.

Finally, we used the universal pretrained HiFi-GAN vocoder introduced by Kong et al. (2020) to synthesise waveforms from the mel-spectrograms predicted from both TTS and SAC. We discussed the HiFi-GAN vocoder in more detail in Section 2.

7.4.3 Pronunciation correction evaluation

7.4.3.1 Listening test design

To evaluate SAC's pronunciation correction capability we ran an AB listening test in which listeners were presented with pairs of samples (each synthesised under a different condition), and asked to choose which pronunciation was preferred; a 'no preference' option was also available. In order to answer H_1 and H_2 we synthesised samples under the following four conditions:

- TTS_G: Transformer TTS using grapheme inputs
- SAC_G: SAC using grapheme inputs only
- SAC_{US}: SAC using grapheme and US female speech code inputs
- SAC_{Scot}: SAC using grapheme and Scottish female speech code inputs

From these we created all 6 possible pairings of conditions, to be presented to listeners. Each synthesised sample consisted of a target word placed within the carrier sentence *'How is ... pronounced?'*. For each condition we used the same set of 78 target words to synthesise samples. Each of the target words were manually identified as: not in LJ-train, mispronounced by SAC_G, and spoken by at least one US and one Scottish female non-target speaker. For SAC_{US} and SAC_{Scot} the target word was represented by speech codes extracted from a randomly-chosen VCTK speaker of that accent. For SAC_{US}, the speaker was chosen from amongst the 18 available US female speakers in VCTK (with each speaker on average contributing 4.3 words), and for SAC_{Scot} from amongst the 13 available Scottish female speakers (on average contributing 6 words each). All samples used in the listening test can be found on the samples page[3]. Each stimulus within the listening test comprised a pair of samples, each generated under a different condition. In total there were 468 stimuli in the test: 6 condition pairs $\times$ 78 stimuli = 468 stimuli. Since this amount was too many for a single listener to judge, we split them into 6 subtests using a Latin square[4]. Each such subtest took a listener approximately 15 minutes to complete. This design ensured that each subtest contained all 6 condition pairs and all 78 target words. To mitigate ordering effects, we randomised each subtest's stimuli order and each within-stimulus condition order, differently per

[3] https://jonojace.github.io/IS22-speech-audio-corrector
[4] https://cs.uwaterloo.ca/~dmasson/tools/latin_square

listener.

We used Prolific[5] to recruit 90 listeners for our listening test, all of whom were native English speakers and US citizens: this was to test H_2, where we expected to generate more accurate US English pronunciations when using speech codes from a US-accented non-target speaker than a different accent. Each of the 6 subtests was implemented with Qualtrics[6] and was taken by 15 listeners.

7.4.3.2 Bradley-Terry statistical analysis

We used Bradley-Terry model (Bradley and Terry, 1952) parameter estimation to obtain scores representing the relative strength or ability of each condition. We estimated the scores s_i using a set of simultaneous equations induced by each pair of conditions i and j, defined by Equation 7.2. We calculated probabilities of pairwise comparisons using a wins matrix $W_{i,j}$ and estimated s_i for each system. We adjusted for the 'no preference' option by assigning half a 'win' to each condition. We estimated the Bradley-Terry parameters from $W_{i,j}$ using the Iterative Luce Spectral Ranking algorithm[7].

The scores s_i are non-negative and can be thought of as measures of an entity's ability. If $s_i = s_j$, the entities are equally likely to win a comparison. If s_i is much greater than s_j entity i is much more likely to win. The scores s_i are typically estimated from data using maximum likelihood estimation. Given a set of pairwise comparison outcomes, the likelihood function is constructed, and the scores are adjusted to maximise this likelihood.

$$P(i > j) = \frac{s_i}{s_i + s_j} \tag{7.2}$$

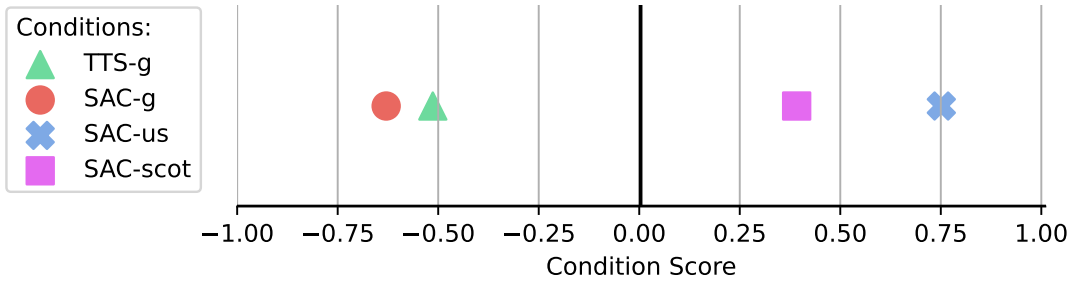

Figure 7.2: A visualisation of the Bradley-Terry model parameters estimated by maximum likelihood via the Iterative Luce Spectral Ranking algorithm. These parameter estimates can be regarded as the score or strength of each condition (*higher* is better) and are calculated from the pairwise comparison count matrix in Table 7.1. Precise score values are -0.63 for SAC_G, -0.51 for TTS_G, 0.39 for SAC_{Scot} and 0.75 for SAC_{US}.

7.5 Results

7.5.1 H_1: speech codes enable one-off pronunciation correction

In Figure 7.2 we observe *considerably* higher scores for SAC_{US} and SAC_{Scot} than for TTS_G and SAC_G. The 'Combined' column in the counts matrix presented in Table 7.1 shows that both SAC_{US} and SAC_{Scot} are preferred approximately twice as frequently as the grapheme-input conditions. Consequently, we can conclude that one-off pronunciation correction using speech codes is successful, supporting H_1. We also observe that TTS_G is slightly preferred over SAC_G. Expert listening (by the first author) revealed that TTS_G pronounces 7 of the target words correctly, while all 78 target words are *by design* mispronounced by SAC_G.

7.5.2 H_2: US speech codes lead to more preferred pronunciations than Scottish ones

We find that SAC_{US} is preferred over SAC_{Scot} by our US listeners, supporting H_2. To further investigate this result we determined, by expert listening, the proportion of target words that were mispronounced when synthesised by either SAC_{US} or SAC_{Scot}. We found that SAC_{Scot} was more likely to incorrectly pronounce words,

[5] https://www.prolific.co/
[6] https://www.qualtrics.com/
[7] http://choix.lum.li

Table 7.1: Count matrix obtained from subjective listening tests taken by native US participants. The i,j-th count is the number of times condition i is preferred over condition j. Ties are accounted for by assigning half a win to each item.

	TTS_G	SAC_G	SAC_{US}	SAC_{Scot}	Combined
TTS_G	-	604.5	263	346	1213.5
SAC_G	565.5	-	215	315.5	1096
SAC_{US}	907	955	-	674.5	2536.5
SAC_{Scot}	824	854.5	495.5	-	2174

mispronouncing 19 of the words (24% of all words) versus SAC_{US} mispronouncing 12 words (15% of all words). This difference in robustness may be due to Scottish speech codes being from a different data distribution than the US speech codes encountered during training. Nevertheless we believe that Scottish speech codes worked reasonably well, exhibiting pronunciation performance superior to grapheme input, and do not noticeably affect the identity of the synthesised voice.

Furthermore we identified several words where SAC_{Scot}'s pronunciation is correct but SAC_{US}'s rendition was preferred by listeners simply because SAC_{Scot} pronunciations are Scottish. For example SAC_{US} pronounces 'derby' as /ˈdɝː.bi/ but SAC_{Scot} pronounces it as /ˈdɑː.bi/ which would be judged as incorrect by most native US speakers. Other such target words include 'mobile', 'bother', and 'comedy'.

7.6 Practical considerations

7.6.1 Building and maintaining a speech-based pronunciation lexicon

Our approach requires a speech-based pronunciation lexicon mapping wordtypes to speech examples. In this chapter's experiments, we constructed a lexicon from the popular open source dataset LJ Speech which consists of *read* speech. To some extent, this simulates using found data to construct a lexicon. However, if real found data were used instead, it could potentially lead to inferior pronunciations due to words being pronounced within more natural and fluent speech. In such

cases, there might be greater coarticulation with adjacent words or even word reductions. Moreover, this coarticulation could pose challenges for the forced alignment model in accurately extracting precise word boundaries, potentially resulting in the inclusion of phonemes from neighbouring words.

In a practical production setting, creating a speech-based pronunciation lexicon may be done differently. In this setting, requirements of the particular production scenario must be considered first. Example requirements might be:

- Pronunciation corrections should be close to 100% robust. That is, given a pronunciation recording, this *should* result in correct pronunciation when used by the TTS system.
- Pronunciation corrections should be inexpensive. At the very least they should be less expensive than phoneme transcriptions, to obtain, maintain, and use.
- Incorporating corrections on a continuous basis should be possible without additional model training.

Taking into account the above requirements we propose a procedure for developing a voice for a new language that would involve the following steps:

1. Training the TTS system using a high-quality TTS dataset: Such a dataset comprises high-quality speech recordings and accurate text transcriptions.
2. Identifying mispronunciations: Once the TTS system is trained, an evaluation is conducted to identify instances where mispronunciations occur in the synthesised speech output. This evaluation could involve expert listening or automated techniques to detect pronunciation errors.
3. Constructing a pronunciation corpus of speech recordings that cover word-types not in the TTS training data, then extracting word-aligned speech codes from it.
4. Determining overlap in word types between mispronounced words and the pronunciation corpus. These are the word types that would potentially be corrected by using speech codes from the pronunciation corpus.
5. Identifying words not accurately corrected by the pronunciation corpus or absent from it: In this step, attention is focused on words that remain

mispronounced even when the pronunciation corpus is utilised. Additionally, any words that are not included in the pronunciation corpus are also identified.

6. Recording speech examples corresponding to these word types and evaluating whether they correct the mispronunciation.

7.7 Discussion

In this chapter, we evaluated our model using a single target speaker from the LJ Speech dataset. While we have not fully confirmed that our results will generalise to other speakers, it is reasonable to expect that the corrective capabilities of the Speech Audio Corrector (SAC) will be effective to some extent across different speakers. Although we could have increased the experimental rigour by including evaluations of SAC trained on additional target speakers, we deemed it sufficient for the purposes of this demonstration to focus exclusively on the LJ Speech dataset. This approach allows us to effectively illustrate the method's potential without unnecessary complexity.

7.8 Conclusion

In this chapter, we introduced and demonstrated a novel grapheme-input TTS model called Speech Audio Corrector (SAC). SAC allows for one-off pronunciation corrections using speech as input. Our experimental results indicate that speech from both US and Scottish non-target speakers can effectively correct the pronunciation of a SAC system trained on a single target US speaker. Additionally, we observe that aligning the speech accent with the target speaker enhances the robustness of pronunciation correction. The simplicity of SAC's approach in utilising speech as a source of pronunciation information makes it a cost-effective paradigm potentially suitable for production use cases, particularly in low-resource TTS scenarios where transcribing phoneme-based lexicons is not feasible.

Regarding future work, there are several possible avenues to explore in order to enhance SAC and its applicability in real-world TTS scenarios. In our experiments, although SAC demonstrates promising results, it is not yet completely robust when using accented speech for pronunciation control. One potential solution

could involve leveraging enhanced SSL representations and fine-tuning SSL models to remove accent information, which may help address this limitation. Another solution could entail using speech codes from non-target speakers during training.

Additionally, SAC holds potential for application in multilingual code-switching scenarios within grapheme-input TTS. By utilising speech codes, foreign language words can potentially be pronounced more accurately than relying solely on their spellings. This opens up possibilities for SAC to support a wider range of languages and facilitate smoother transitions between languages during synthesis.

Lastly, it is worth noting that SAC currently relies on high-resource Automatic Speech Recognition (ASR) systems to generate word alignments, which are used to construct the speech-based lexicon. In future work, it would be valuable to explore alternative approaches that reduce or eliminate the dependence on such high-resource ASR systems. This could involve investigating methods for word alignment that are more lightweight, making SAC more accessible and feasible for practical deployment in various TTS applications. One potential method involves using word alignments extracted from an end-to-end CTC-based ASR system which is trained from scratch using just speech and text pairs.

Overall SAC, is a promising paradigm that enables low-resource corrections for grapheme-input TTS. This new functionality makes grapheme-input TTS more viable. The model tackles the trade-off between pronunciation control and amount of phonemic resources discussed in Chapter 5. Furthermore it improves on the issue of join artifacts, as observed in Chapter 6, by incorporating speech-based pronunciations on the *input* side to a TTS system, rather than when conditioning a vocoder.

The following chapter details an approach that adopts a slightly different method of using speech-based pronunciations. Rather than directly using speech representations as input to the model, the approach uses speech examples to retrieve grapheme sequences (i.e. spellings) that are closer in pronunciation than the original spelling. This approach does not require training a TTS model from scratch and also produces a pronunciation lexicon (i.e. mapping from original spellings to new spellings) that are more human-interpretable than discretised speech codes. This potentially reduces the cost of lexicon maintenance.

Chapter 8

Correcting text-to-speech mispronunciations using novel spellings generated from spoken examples

This chapter represents an extended version of my research presented in *Spell4TTS: Acoustically-informed spellings for improving text-to-speech pronunciations*, which was published in the proceedings of the Speech Synthesis Workshop 2023 (Fong et al., 2023). This was joint work done in collaboration with Hao Tang and Simon King.

8.1 Introduction

In this chapter, we introduce a novel approach to low-resource pronunciation control: ground-truth spoken pronunciation examples are used to help *discover* spellings that are likely to be better pronunciations than the original spellings when input to a grapheme-input TTS system.

This research was motivated in part by limitations of the Speech Audio Corrector (SAC) method that were discussed in Chapter 7. Pronunciations formed from self-supervised learning (SSL) speech codes may still include non-segmental information such as prosody (Yang et al., 2021; Li et al., 2023), reducing their effectiveness

as reliable stand-ins for pronunciation (van Niekerk et al., 2022). Additionally, although SSL-based pronunciations are likely more affordable to commission than phoneme-based ones, generating them relies on large pretrained SSL speech models such as wav2vec 2.0 (Baevski et al., 2020) or HuBERT (Hsu et al., 2021), which are computationally expensive to train and require large amounts of speech data. Furthermore, due to the difficult-to-interpret nature of SSL speech codes, maintaining a lexicon of them is difficult. The speech code sequences are not easy to understand at a glance and consequently it is not clear how to edit them by hand. These limitations potentially reduces SAC's usefulness for low-resource languages.

An alternative to using speech code or phoneme-based pronunciations is to simply use *grapheme*-based ones. Graphemes are a human-parsable interface to pronunciation. One *could* address TTS-mispronounced words by requesting annotators to search for alternative spellings that the TTS system pronounces correctly. However, this search is a trial-and-error process for languages whose mapping from graphemes to speech sounds is ambiguous. This process is time-consuming and difficult as annotators must be able to internalise how the TTS model pronounces grapheme sequences in order to make accurate predictions. Furthermore, since a TTS model's pronunciation of grapheme sequences may change with each retraining, depending on its hyperparameters, weight initialisation strategy or training data, using grapheme-based pronunciations may necessitate frequent manual re-transcription. This could potentially result in higher costs compared to using phoneme-based pronunciations.

In this chapter, we introduce a novel *automatic* approach to discovering a spelling that – when input to a grapheme-based TTS model – results in a pronunciation close to a ground-truth spoken example. These spellings may be stored in a simple dictionary, to be used during synthesis instead of that word's original spelling. This automatic approach, named Spell4TTS, has the potential to reduce or even eliminate the pain points of transcribing grapheme spellings manually. The approach works as follows: Firstly, for any given wordtype, we obtain a time-aligned spoken example, and use an ASR system to generate many candidate spellings (in the order of hundreds or thousands, for example). In our work we generate these spellings from the ASR system's n-best list. Secondly, we synthesise all candidate spellings using the pre-existing target TTS system. This is because

we wish to know how the TTS system pronounces these plausible alternative spellings. Finally, we generate an acoustic distance between the rendition for each candidate spelling, and the ground-truth spoken example. It is important for the acoustic distance to reflect desirable pronunciation features such as phoneme class, while disregarding undesirable ones such as speaker identity. The acoustic distance is used to rank all of the candidate spellings. A final 'good' spelling can then be chosen in one of two ways, either i) we select the top ranked spelling if a fully automatic approach is desired, or ii) we use a Human-in-the-Loop (HitL) to select what they judge to be the closest in pronunciation to the spoken example from a shortlist of the top ranked spellings. This HitL task could be performed by any native speaker of the language.

Our experimental results strongly suggest that this is a viable method for generating interpretable pronunciations. We provide results for a variety of acoustic distances, including a novel one using self-supervised (SSL) speech representations. The method requires grapheme- or WordPiece-based ASR and TTS models, plus natural spoken examples of the words for which improved spellings are required. These resources are all fairly straightforward to construct, and none of them require a phonemic pronunciation dictionary.

Our method is distinct from spelling reform efforts (Zachrisson, 1931; Hodges, 1964), as it is not intended to generate *simplified* versions of original spellings for human use. In fact, our method makes no use of the original spelling of a word, but generates a new spelling based solely on a spoken example; this typically results in a different spelling (e.g., the word originally spelled "Marseilles" is spelled "marssayy" by our method). A core advantage of our method is that it is applicable to any existing TTS grapheme-based system and does not require any changes or further training of that system.

We believe that our method has the potential to impact the development of TTS voices when pronunciation resources are prohibitively expensive to obtain. The method creates a grapheme-based pronunciation dictionary that is more intuitive and easier to interpret than a speech-based one, forgoing the need for specially trained linguists and subsequently reducing the barrier to entry for TTS voice development. Passionate users of the technology *themselves* may be able to provide their own speech recordings and help discover 'good' spellings, scaling TTS voice development and maintenance far beyond the confines or longevity of a

single team. Furthermore, we believe the method has the potential to rejuvenate grapheme-input TTS research for low-resource languages.

In this chapter we show the following results: We demonstrate that our automatic transcription method Spell4TTS can discover grapheme-based pronunciations using speech obtained from spoken examples. We also demonstrate that it will likely work with a variety of traditional or SSL speech features, and that it can incorporate HiTL filtering to further improve the quality of discovered pronunciations.

8.2 Background

8.2.1 Automated pronunciation mining from audio

In this chapter we define Automated Pronunciation Mining as improving or discovering pronunciations from speech in an automatic fashion.

Prior work has developed tools for building phoneme-based pronunciation lexica from speech, for instance the 'lex4all' project (Vakil et al., 2014) which used the Salaam method (Rosenfeld and Sherwani, 2004). This method used an existing ASR system built for English to transcribe English phonemes from the speech of a variety of non-English target languages. These English phoneme sequences were then used to represent the non-English word pronunciations. An advantage of this approach is that no phoneme set or mapping is required in the target languages. However, a limitation of this approach is that the inventory of English phonemes cannot represent all potential pronunciations found in various target languages. Consequently, English ASR systems may struggle to accurately capture and encode all speech sounds in certain languages.

Another line of work by Sun et al. (2023) has used speech examples to update a neural sequence-to-sequence TTS frontend. In their work, the frontend predicts linguistic features from input raw text, with the linguistic features being those generated by a more traditional pipeline-based frontend. To generate linguistic features to update the neural frontend they use the traditional pipeline frontend to decode the linguistic features from spoken examples that contain out of vocabulary words, and then update the neural front-end with the resulting <text, linguistic features> input-output pair. This work demonstrated that updating the neural frontend in such a way increased the accuracy of pronunciations predicted by the

frontend for out-of-vocabulary words. One downside of this approach however is that it requires a pre-existing pipeline based front-end. These traditional frontends are notoriously hard to build and maintain since they consist of a suite of machine learning models and heuristics that depend on both labelled data and trained experts. In contrast, in this chapter we focus on purely grapheme-input TTS that requires no specialist linguistic frontend outside of text normalisation.

Finally, there has been a line of research that has investigated whether a DTW-based distance in acoustic feature space can be used to compare and quantify the difference between native and non-native accents. For example, Bartelds et al. (2020) attempted to assess foreign accent strength in North American English by comparing the speech of native and non-native speakers. They found that their acoustic distance, calculated using DTW between two sequences of MFCCs, has a strong correlation with human judgements of whether speech seems native-like. Additionally, they show that their acoustic measure is not just sensitive to phonetic differences but also to *intonational* and *durational* differences (which would be relevant for tonal and quantity languages respectively). Consequently, in our experiments, we also explored calculating acoustic distances in SSL speech representation space, aiming to mitigate the influence of these variations arising from prosody differences.

8.2.2 Grapheme-based pronunciations

In the domains of linguistics, speech science, and speech technology, the conventional approach to transcribing and representing word pronunciations involves the use of phonemes (Alharbi et al., 2021; Ning et al., 2019). Phonemes possess advantageous characteristics for certain applications such as being standardised and representing meaningful differences between the sound patterns in language (i.e., contrastive function) (Swadesh, 1934). However, a drawback of phonemes lies in their lack of *accessibility*, as they necessitate formal linguistic training for comprehension. In response to this limitation, alternative non-phonemic methods have been developed, employing grapheme-based systems for pronunciation transcription that offer increased accessibility to the general population (Walker, 1830). These systems utilise the character sets specific to each language and transcribe speech sounds using sequences of graphemes that commonly and predictably correspond to those sounds. For instance, in English, the pronunciation

of “Champagne” could be represented as “sham·pain” rather than the phonemic transcription /ʃæmˈpeɪn/.

Phonics serves as a prominent example of grapheme-based pronunciations in practical application (Goodman, 1993). It is an instructional approach widely employed in educational settings, particularly in schools and preschools, to facilitate the development of reading and spelling skills. The foundation of phonics lies in the “alphabetic principle”, a fundamental concept that states that there exist systematic and predictable associations between letters (or letter sequences) in written language and corresponding speech sounds (Liberman et al., 1989). This principle enables individuals to predict the spelling of unfamiliar words or decode unfamiliar spellings. However, it is worth noting that the efficacy of this approach can be compromised in languages such as English or French, which possess highly opaque orthographic systems (Katz and Frost, 1992; Marjou, 2019). Historical influences, including shifts in pronunciation over time and the incorporation of loanwords from other languages, contribute to the potential inaccuracies associated with the application of the alphabetic principle in these languages (Campbell, 2013).

Another example of grapheme-based pronunciations can be observed in the efforts of language spelling reform, as observed in languages including English, French, German, and Dutch (Zachrisson, 1931; Ball, 1999; Geerts et al., 1977). Spelling reform efforts are deliberate and systematic alterations to a language’s orthographic system, driven by specific objectives. One prevalent aim is to enhance the consistency and simplicity of spellings, thereby facilitating ease of learning and utilisation. For instance, irregular spelling patterns, diacritics, and ‘special’ characters (such as ‘ß’ becoming ‘ss’ in the Swiss and Liechtenstein forms of written German) can be standardised and simplified.

Similar to spelling reform efforts, employing graphemes as a representation of pronunciation could potentially facilitate the development of Text-to-Speech (TTS) voices. This approach is particularly promising for low-resource settings, because untrained users could more easily contribute to or modify spelling-based pronunciations themselves, ultimately enhancing the systems they rely on. By empowering individuals to provide or edit pronunciation information based on graphemes, TTS voice development becomes more accessible and inclusive.

8.2.3 Goodness of pronunciation algorithm

The Goodness of Pronunciation (GOP) algorithm, first introduced by Witt (2000), is a technique used in speech processing and language learning to evaluate the pronunciation quality of spoken words or phrases. It is particularly useful in automated language learning systems, speech therapy applications, and accent reduction programs, providing a score that reflects how closely a learner's pronunciation matches that of a native speaker or a reference pronunciation. The GOP algorithm achieves this by recording a learner speaking a word or phrase and comparing it to the reference pronunciation. It breaks down the spoken word into individual phonemes, calculates scores for each sound based on their match to the reference using probabilities from an ASR system, and provides feedback, highlighting well-pronounced sounds and areas needing improvement. In contrast, our approach avoids the need for phoneme transcriptions, potentially making it more scalable across the world's languages.

8.3 Proposed method: Spell4TTS

Spell4TTS is our proposed method for automatically deriving word spellings from spoken examples. The three stages are: candidate spelling generation, candidate synthesis, and candidate ranking, as illustrated in Figure 8.1. In the rest of this chapter, we provide concrete solutions for each stage. The overall method does *not* depend on these details and alternative approaches could easily be proposed for any of the stages.

8.3.1 Stage 1: Generate candidate spellings

This stage needs to propose a set of candidate spellings, amongst which there is at least one spelling that, when synthesised, will closely match the spoken ground-truth example. One option might be to propose *all* possible spellings, but that would be impractical, and most of them would be irrelevant. Another option would be to perturb the original spelling by some means, creating many plausible variants, but without phonemic pronunciation resources the means of doing this are not obvious, and this approach risks not finding a good spelling that is very different from the original. Another approach would be to use an ASR system to generate candidate spellings. This is the approach that we use in this work, as

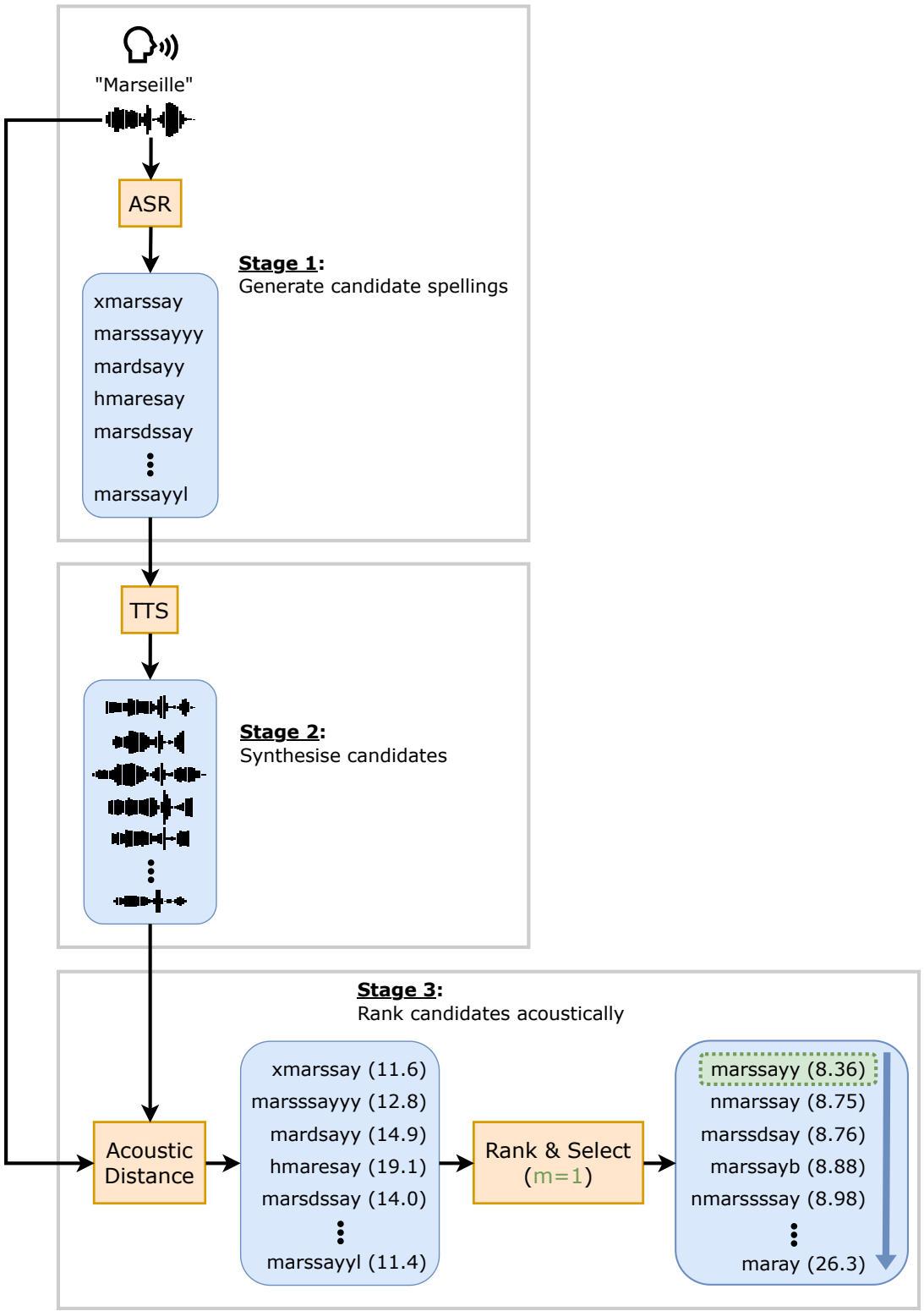

Figure 8.1: Overview of Spell4TTS, a method for automatically finding word spellings from spoken examples, for improving TTS pronunciation. Pictured are actual candidate spellings and acoustic distances from AcousticRank detailed in Section 8.4.7.

only generating the spellings that are *acoustically-related* to the spoken example reduces the complexity of Stage 2 and Stage 3. We used a simple ASR system to generate a long n-best list of hypotheses from the spoken ground-truth example.

8.3.2 Stage 2: Synthesise candidates

All candidates from Stage 1 are synthesised using the particular TTS system for which we wish to improve pronunciation.

8.3.3 Stage 3: Rank candidates acoustically

The acoustic distance between each synthesised candidate and the spoken ground-truth example is measured, and used to rank the candidates. In the fully-automated version of our method, the top-ranked candidate is chosen. If there is an optional Human-in-the-Loop (HitL), they will choose amongst the top few candidates, by listening.

8.4 Experiments

Given the many design choices possible for the three stages, we limit the current work to testing the following three hypotheses:

H1 – calculating acoustic distances using representations from self-supervised learning (SSL) will identify better-pronounced synthetic speech than when using MFCCs, since prior work has shown that SSL speech representations are able to separate phonemic information from channel, speaker, and other unwanted properties. Testing this hypothesis first will allow us to employ the best acoustic distance measure when testing subsequent hypotheses.

H2 – the proposed method will find a spelling that results in a more accurate pronunciation than either the original spelling or the 1-best spelling from the ASR baseline (described shortly in Section 8.4.4), because the method uses acoustic information to choose the spelling from the list of acoustically-motivated candidates proposed by ASR. This is the main claim of the current work.

H3 – placing a Human-in-the-Loop in Stage 3 will find better-sounding spellings than the fully automatic method. This is a secondary claim.

8.4.1 Speech dataset & word-aligned spoken examples

We use the LJSpeech dataset to train our models. However, unlike the conventional TTS use case of the dataset, we partitioned it into two equally sized halves: $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$. The former was employed for training the ASR and TTS models, while the latter was used to obtain a sizeable number of mispronounced out-of-vocabulary wordtypes for evaluating our method. $\mathcal{D}_{\text{test}}$ was formed by taking utterances that contain the lowest frequency wordtypes one-by-one until it consisted of half of the utterances of LJSpeech; the remaining utterances were placed in $\mathcal{D}_{\text{train}}$. Consequently, $\mathcal{D}_{\text{test}}$ contains 13811 wordtypes, out of which 8343 were absent from $\mathcal{D}_{\text{train}}$. $\mathcal{D}_{\text{train}}$ contains 5657 wordtypes. In order to ensure that all ASR hypotheses would be valid candidate spellings the normalised transcript of LJSpeech was further processed to contain only lowercase alphabetic characters without punctuation. We removed punctuation in order to ensure that the ASR model would not generate them. To acquire spoken examples for each wordtype in $\mathcal{D}_{\text{test}}$, we used the Montreal Forced Aligner[1]. It is worth noting that although we adopted this high-resource word alignment solution, our method is adaptable to use a lower-resource solution, manual segmentation, or simply a purpose-recorded isolated word speech corpus.

8.4.2 Use of non-autoregressive transformer-based text-to-speech model

For the experiments in Chapter 7 we adopted the autoregressive sequence-to-sequence model named 'Transformer TTS'. However in this Chapter, with the goal of faster experimental turnaround, we instead opted for a popular implementation of non-autoregressive transformer-based TTS named 'Fastpitch' (Łańcucki, 2021). One drawback of autoregressive sequence-to-sequence-based TTS models such as Tacotron (Wang et al., 2017b) and Transformer TTS (Li et al., 2019) is their slow training and inference speed. The decoder in these models operates in an autoregressive manner, generating one frame of output at a time (e.g. a mel-spectrogram frame), conditioned on previously predicted frames and aligned with the encoder outputs. In contrast, feed-forward Transformer-based TTS models have gained popularity due to their full parallelisation capability, resulting in significantly faster training and inference while maintaining competitive naturalness in terms

[1] https://montreal-forced-aligner.readthedocs.io

of subjective MOS ratings.

FastSpeech was introduced as one of the first fully parallelisable approaches, achieving an inference speedup of approximately 270 times over the autoregressive Transformer TTS. It uses an autoregressive teacher model to extract durations and employ knowledge distillation. Durations are obtained from the decoder-to-encoder attention alignments of the trained teacher model, specifically from the attention head with the most diagonal attention matrix. These extracted durations are used to train the duration predictor, and during testing, predicted durations are used to upsample the encoder outputs to match the desired frame-rate.

The weakness of FastSpeech, however, lies in the requirement of first training an autoregressive teacher model to extract durations, which is a complex and time-consuming step. Additionally, the extracted durations are not as accurate as those from a purpose built aligner resulting in lower naturalness after model training (Ren et al., 2020). To address this, FastSpeech 2 (Ren et al., 2020) and FastPitch (Łańcucki, 2021) were introduced. They circumvent the need for a teacher model to extract durations and instead leverage "ground-truth" durations obtained from a forced aligner. During training, these ground-truth durations are employed to train a duration predictor and to upsample the encoder outputs to the frame-rate. During inference, the duration predictor's outputs are used for upsampling.

In this work, we utilise FastPitch due to its robust implementation by NVIDIA[2] and its demonstrated effectiveness with grapheme inputs (Łańcucki, 2021). To simulate a scenario with limited resources and unavailable ground-truth alignments, we employ an extension to FastPitch called "monotonic alignment search" (Kim et al., 2020). This method eliminates the need for ground-truth alignments by iteratively searching for the most statistically probable alignment using dynamic programming during training. An initial diagonal alignment is assumed at first and then gradually refined. Durations are then extracted from the alignment and used for upsampling encoder outputs and training the duration predictor.

[2] https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/SpeechSynthesis/FastPitch

8.4.3 Calculating the acoustic distance between synthesised candidate spellings and ground-truth spoken examples

Our proposed method uses an acoustic distance to quantify the pronunciation difference between a ground-truth spoken example of a word, and alternative candidate spellings synthesised by the target TTS system. In our experiments, for the acoustic distance we adopt dynamic time warping (DTW), introduced by Bellman and Kalaba (1959), to account for the differences in length between ground-truth spoken examples and their corresponding synthesised candidate spellings. While other measures exist for quantifying the difference in pronunciation between two speech samples, for example the goodness of pronunciation measure uses phoneme ASR posterior probabilities (Kanters et al., 2009; Sheoran et al., 2023), we adopt DTW as it is a simpler method which does not necessitate a pre-existing phoneme-based ASR system.

This is also known as pronunciation accuracy, and it is often calculated as the ratio of the number of correctly detected phonemes to the total number of phonemes in the test speech signal.

Dynamic time warping is a common method used when a measure of similarity is required between two temporal sequences, which have different lengths or are temporal distortions. For example, given a speech recording of the word 'cat' spoken twice, once fast and once slow, DTW can be used to calculate a similarity score that is robust (i.e. roughly invariant) to the speed difference between the two renditions.

DTW finds the optimal alignment between two sequences of vectors by warping the shorter one to the length of the longer one. This warping is not done in a simple linear fashion, but rather considers all possible warpings or alignments between the two sequences. An optimal alignment in DTW is the alignment path that minimises the total distance (or cost) between the sequences. This path ensures the best possible match between elements of the sequences by allowing non-linear adjustments, accommodating variations in speed or length, and providing the lowest cumulative cost according to the chosen distance metric.

The calculation of the optimal alignment that minimises the total cost is made computationally efficient via dynamic programming. To compute the local dissimilarity between a pair of vectors belonging to two timesteps (one from each

sequence), a local cost must be used. Common examples are Euclidean distance, or cosine similarity. When using Euclidean distance in the mel-cepstrum domain along with DTW this is known as mel-cepstral distortion DTW (MCD-DTW). This is a common objective metric for evaluating speech technology systems by comparing synthetic output, for example TTS or voice conversion output, with some ground-truth speech. For each pair of timesteps, Euclidean distance is calculated between two vectors of mel-cepstrum coefficients; the optimal path with minimum cost is then found using DTW. MCD-DTW is the main inspiration behind what we call more generally 'acoustic distance' in this chapter. We use an acoustic distance using MFCCs, and also one calculated in SSL speech representation space, where cosine similarity rather than Euclidean distance is adopted, because SSL speech representation systems such as HuBERT and wav2vec2.0 are trained using cosine similarity to output encodings close to discretised vector targets.

8.4.4 Models

- Automatic Speech Recognition (ASR): We utilised a CTC end-to-end ASR architecture based on the work of Graves et al. (2006) to generate candidate spellings from spoken examples. The 'CRDNN' architecture (Convolutional Recurrent Deep Neural Network) comprised 2 CNN blocks with 128 and 256 channels, a 3 by 3 kernel size, and no time pooling or subsampling factor. These were followed by two Bidirectional LSTM layers with 512 hidden units and then two DNN layers, with 512 hidden units and 28 output units (26 alphabet characters, whitespace, and the blank token). For each input spoken example we generated 1000 candidate spellings using CTC beam search decoding with a 1000-best list, beam size of 2000, and beam threshold of 50. To prevent generation of the whitespace token, we force its probability to zero for all timesteps. In this work, we refrained from using an encoder-decoder ASR architecture with top-k or nucleus sampling, or language model rescoring, since these techniques might overpower the acoustic model, making it less likely to generate novel spellings. We used the SpeechBrain toolkit[3] to implement the model, and trained it from scratch on 90% of $\mathcal{D}_{\text{train}}$, with 10% left for validation, picking the checkpoint with

[3] https://speechbrain.github.io

the lowest WER over the validation set.

- Text-to-speech (TTS): We used a grapheme-based FastPitch TTS model to synthesise these candidate spellings, adopting the same architecture as the original work by Łańcucki (2021). We trained the FastPitch model for 1000 epochs using a batch size of 16 on $\mathcal{D}_{\text{train}}$, with monotonic alignment search (Kim et al., 2020) to obviate the need for external alignments. For waveform generation, we utilized the readily available universal HifiGAN vocoder as introduced by Kong et al. (2020). This vocoder is designed to convert mel-spectrograms into waveforms. While there are alternative methods like Griffin-Lim, we opted for HifiGAN to optimize clarity, particularly for the purpose of our subjective listening tests.

- Self-supervised speech representations (SSL): We investigated calculating acoustic distances between SSL speech representations derived from a HuBERT model[4] (Hsu et al., 2021) implemented by the authors of van Niekerk et al. (2022). This implementation can optionally extract 'soft' speech representations, which were claimed to better capture the nuances of pronunciations for voice conversion purposes.

8.4.5 Evaluation & statistical analysis

We evaluate the pairs of conditions in Experiment 1 and 2 using subjective AB listening tests. We do not employ objective metrics as they may neglect more nuanced aspects of pronunciation, such as syllable stress and coarticulation.

8.4.5.1 Listening test stimuli

We selected 100 orthographically opaque wordtypes for Experiment 1 and another 100 for Experiment 2 in the following way: First, to identify the wordtypes in $\mathcal{D}_{\text{test}}$ that would most likely be mispronounced by our TTS model we used the Phonetisaurus G2P model, introduced by Novak et al. (2016), trained on $\mathcal{D}_{\text{train}}$. We calculated the phone-error-rate (PER) of each wordtype in $\mathcal{D}_{\text{test}}$ according to ground-truth pronunciations in CMUDict (CMUdict, 2014), and then selected the 200 highest PER wordtypes over 7 characters long. Spoken audio for these wordtypes is then retrieved from $\mathcal{D}_{\text{test}}$ and is used in all the conditions except

[4] https://huggingface.co/facebook/hubert-base-ls960

OriginalSpelling to find spellings which are then synthesised by our TTS model. The G2P model was trained on a set of <wordtype-phoneme sequence> pairs created by looking up each $\mathcal{D}_{\text{train}}$ wordtype in CMUDict. All wordtypes, spellings, and synthesised stimuli can be found on the samples page[5].

8.4.5.2 Listening test design

For each experiment we generated 100 stimuli from each of the 5 conditions belonging to the experiment. We then paired the conditions together making 10 pairs of conditions which altogether formed 1000 AB questions. Since 1000 questions were too many for any listener to rate in a single session, we broke them down into 10 subtests. Each subtest contained 100 AB questions and we used a Latin square to ensure that each subtest contained 10 AB questions from each pair of conditions. For each AB question participants listened to the spoken example of a wordtype, and two synthesised spellings. Both the question order and order of A and B were randomised on a per participant basis. They were prompted to select the synthesised rendition that most closely matches the spoken example in terms of pronunciation. The prompt was as follows: "Which of the following two renditions of <*wordtype*> are closer to the reference recording in terms of pronunciation?". They were also allowed to select a third 'no preference' option. We recruited 30 native English-speaking participants from North America using Prolific[6]. We selected North American participants since the speaker of the LJ Speech dataset is American. Each of the 10 subtests were taken by 3 participants.

8.4.5.3 Statistical testing

To determine significance of our listening test results we performed pairwise proportion z-tests to determine whether two proportions are different according to their sample sizes. In our case, for each pair of conditions we obtained 10 observations from each of the 30 participants, giving 300 in total.

In an identical fashion to 7.4.3.2 we applied the Bradley-Terry model (Bradley and Terry, 1952) to obtain scores that reflect the relative strength of each condition based on pairwise comparison data.

[5] https://spell4tts.github.io/samples
[6] https://www.prolific.co

8.4.6 Experiment 1 : Determining the features for calculating an acoustic distance (tests H1)

Experiment 1 tests H1 and therefore selects the acoustic distance measure to use in Experiment 2. An effective and useful measure should enable the comparison of two renditions of a word, demonstrating sensitivity to phonetic variation and syllable stress while remaining insensitive to other variations in speech. To handle variations in sequence length, different approaches are employed for continuous and discrete representations. For continuous representations, Dynamic Time Warping (DTW) is utilized, with the option of using either cosine or Euclidean local distance measures. In contrast, for discrete representations, the edit distance metric is applied. Two types of speech features were extracted from synthesised candidates obtained in Stage 2, i) MFCC features, and ii) SSL speech representations from layer 7 of the HuBERT model, which have been shown to perform well on phone discrimination tasks (Baevski et al., 2020; Nguyen et al., 2020; van Niekerk et al., 2021). We include 4 varieties of SSL speech representations that have each been shown to exhibit different properties. In total, we establish five conditions that are used to automatically identify the good spelling for each spoken example in Experiment 1:

- MFCC (Euclidean+DTW): MFCCs are a traditional feature type commonly used in ASR systems and are engineered to filter out speech information such as pitch. We extracted the first 12 MFCCs using the Librosa package[7].
- HUBERT-RAW (Cosine+DTW): Raw 768-dimensional HuBERT representations are straightforward to extract and have been shown to encapsulate pronunciation, prosodic, speaker identity, and semantic information in a form that is more linearly separable than traditional features like MFCCs as observed by Hsu et al. (2021) and Yang et al. (2021).
- HUBERT-CENTROID (Cosine+DTW): We extracted the centroids, defined as the mean vectors of a cluster, using a 100-cluster k-means model trained on raw HuBERT representations from LibriSpeech-960 (Panayotov et al., 2015). These centroids tend to minimise paralinguistic information like speaker identity, while still retaining features related to pronunciation including place or manner of articulation, as demonstrated by Wells et al. (2022).

[7] https://librosa.org

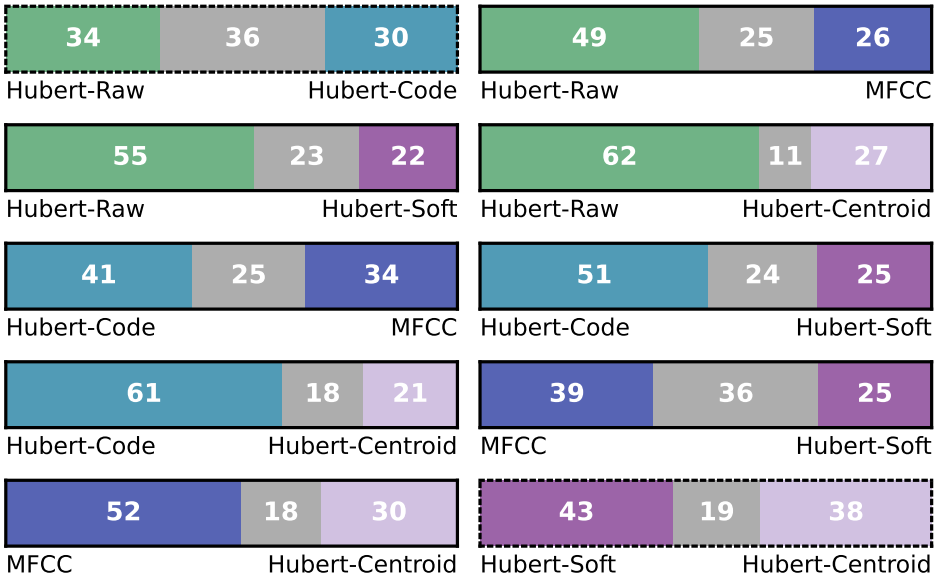

Figure 8.2: Experiment 1 - identifying the features for calculating acoustic distance - listening test results: Proportion plots for the pairs of conditions presented to listeners. The middle grey region indicates 'no preference'. Note: the condition pairs HUBERT-CODE vs. HUBERT-RAW (p-value = 0.18) and HUBERT-CENTROID vs. HUBERT-SOFT (p-value = 0.2) do not exhibit statistical differences, and are highlighted with dotted line edges. All other pairs are statistically different.

- HUBERT-CODE (Levenshtein): We also employ the k-means model's discrete cluster IDs as features because they most aggressively filter out paralinguistic information (van Niekerk et al., 2020; Polyak et al., 2021). This makes them more robust to differences unrelated to pronunciation.

- HUBERT-SOFT (Cosine+DTW): We use soft HuBERT features as a possible improvement over cluster IDs which can sometimes discard linguistic content, leading to mispronunciations in voice conversion (van Niekerk et al., 2022) and TTS as we observed in the preceding chapter. These 256-dimensional soft speech representations are learned to help predict a distribution over discrete speech unit targets. Unlike discrete units, which can sometimes omit pronunciation information, soft units, which model uncertainty, capture more pronunciation information, potentially reducing the likelihood of mispronunciations.

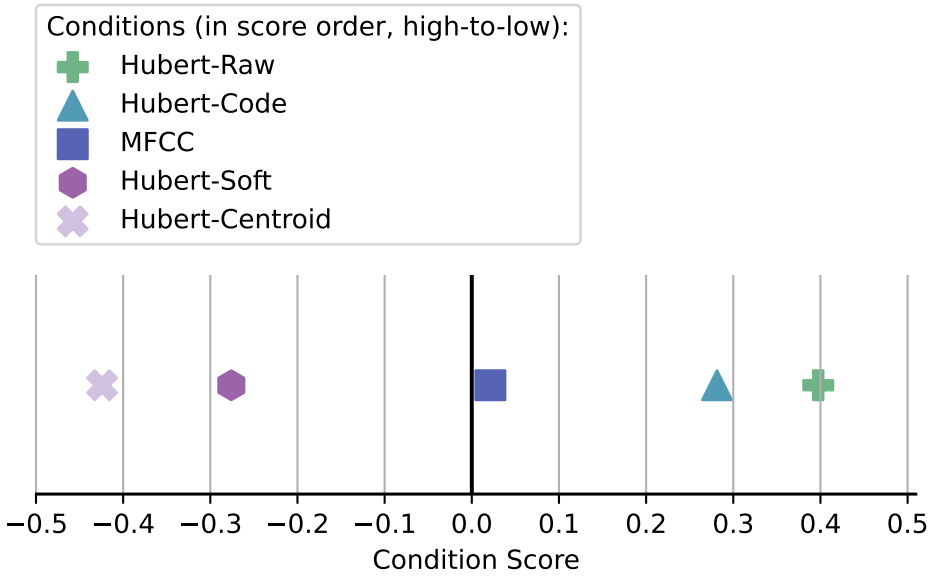

Figure 8.3: Experiment 1 - identifying the features for calculating acoustic distance - listening test results: Condition scores estimated from a Bradley-Terry model. These can be regarded as the quality or strength of each condition (*higher* is better). Score values are 0.4 for HUBERT-RAW, 0.28 for HUBERT-CODE 0.02 for MFCC, -0.28 for HUBERT-SOFT, and -0.42 for HUBERT-CENTROID.

8.4.6.1 Results: H1 is supported - HuBERT features outperform MFCCs for finding pronunciations via acoustic ranking

The listening test results from experiment 1, illustrated in figures 8.2 and 8.3, demonstrate that two types of HuBERT-based features outperform MFCCs. More specifically, HUBERT-RAW performs the best, followed closely by HUBERT-CODE. This finding is interesting because these two conditions are assumed to be on opposite ends of the spectrum in terms of discarding paralinguistic information. It is worth noting that they are not significantly different according to a pairwise proportion z-test (p-value = 0.18). MFCC features perform reasonably well, outperforming both HUBERT-SOFT and HUBERT-CENTROID. Overall our findings suggest that HuBERT features should be used to maximise performance, but MFCCs can be adopted if a simpler pipeline is desired.

8.4.7 Experiment 2: determining the efficacy of our acoustic ranking method versus baseline conditions (tests H2 & H3)

To investigate H2 and H3 we include three conditions which compare our proposed method SPELL4TTS with an ASR baseline and the original spellings, and we also incorporate Human-in-the-Loop refinement with the proposed method and the ASR baseline forming two additional conditions. Each of these five conditions devises a single spelling for each spoken example. These spellings are then synthesised and presented to subjects in a listening test. The following is a list of the five conditions used in Experiment 2:

- ORIGINALSPELLING: The original unmodified spelling.
- ASR: The top-ranked (i.e. 1-best) spelling from the ASR model.
- ASR+HITL: We present m synthesised audios corresponding to the top m highest-likelihood spellings, derived from the ASR hypotheses, to a Human-in-the-Loop (HITL) evaluator. The evaluator then selects the spelling that, in their judgement, best aligns with the pronunciation of the spoken example. For our experiments we set m to 5. Although m could be increased given time and budget, it is likely that large values would be difficult for a human to filter effectively.
- ACOUSTICRANK: This condition uses our proposed method Spell4TTS to retrieve spellings in a fully automated fashion. We calculate acoustic distances using raw HuBERT features due to their superiority in Experiment 1 as discussed in Section 8.4.6.1.
- ACOUSTICRANK+HITL: Same as ACOUSTICRANK but we instead acoustically retrieve the top m candidates which are then refined down to a single one by a Human-in-the-Loop in a similar fashion to ASR+HITL. We set m to 5.

8.4.7.1 Results: H2 is supported - Acoustic ranking outperforms ASR baselines and original spellings

Figures 8.4 and 8.5 both show that the proposed fully automated method, ACOUSTICRANK, is shown to be significantly better than both ASR baselines, indicating

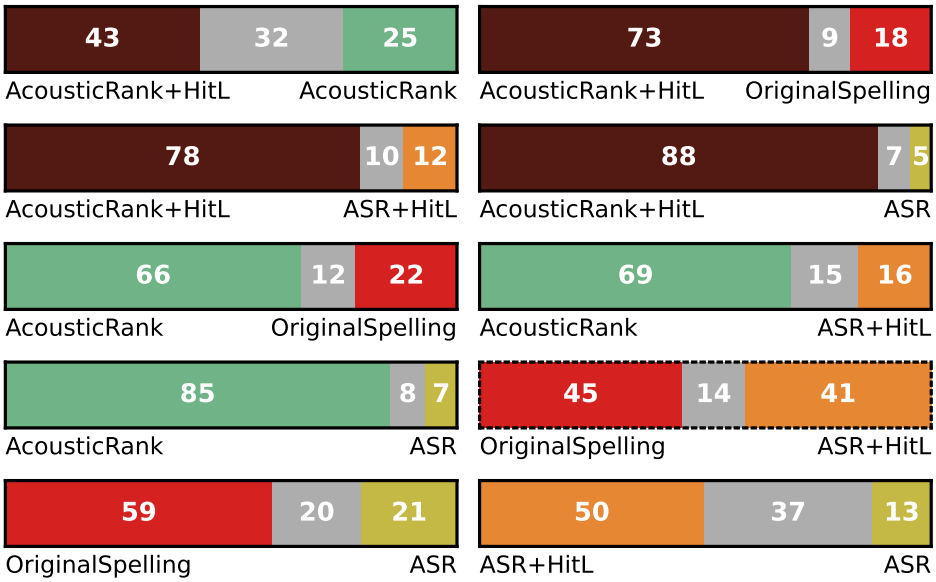

Figure 8.4: Experiment 2 listening test results: Proportion plots. Note: the condition pair ASR+HitL vs. OriginalSpelling does not exhibit a statistical difference (p-value = 0.35), and is highlighted with a dotted line edge. All other pairs are statistically different.

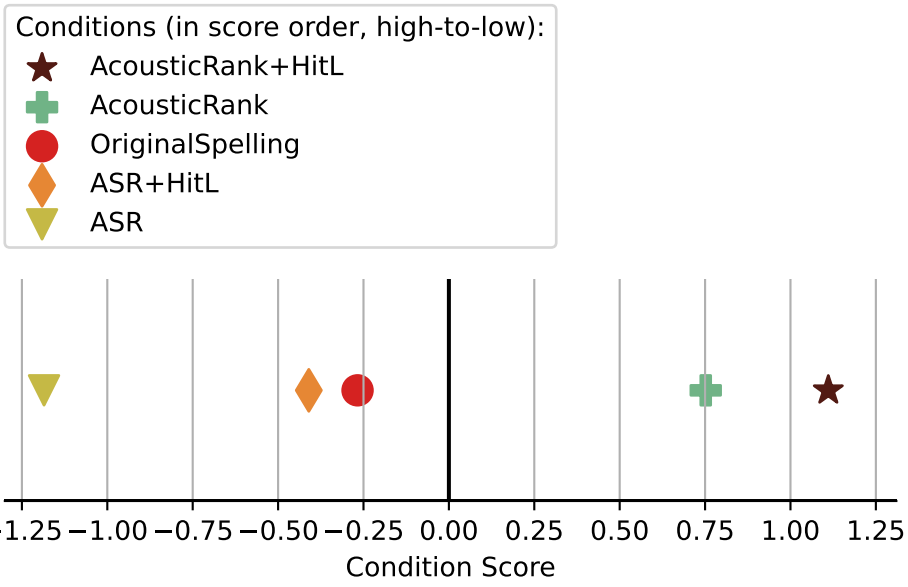

Figure 8.5: Experiment 2 listening test results: Bradley-Terry model condition scores. Score values are 1.1 for AcousticRank+HitL, 0.75 for AcousticRank, -0.27 for OriginalSpelling, -0.41 for ASR+HitL, and -1.2 for ASR. Note that for convenience AcousticRank is assigned the same symbol and colour as the condition Hubert-Raw in Figure 8.3 as it retrieves pronunciations in an identical way.

that better pronunciations often lie outside the ASR 1-best or 5-best lists, and that we can retrieve them automatically. Interestingly, the ASR baselines were found to be no better than using original spellings, even when incorporating Human-in-the-Loop assistance. Notably, ASR performed particularly poorly in comparison to original spellings.

8.4.7.2 Results: H3 is supported - Incorporating a Human-in-the-Loop finds better spellings

Furthermore from Figures 8.4 and 8.5 we observe that incorporating Human-in-the-Loop assistance consistently led to the discovery of better pronunciations and further enhanced the performance of our proposed method. Section 8.4.8.2 delves deeper into this result.

8.4.8 Further analysis

8.4.8.1 Automated retrieval statistics

Figure 8.6 reveals a roughly even distribution of the top-5 spellings retrieved automatically by HUBERT-RAW/ACOUSTICRANK across ASR n-best ranks from 1 to 1000. This evidence, along with the findings presented in Section 8.4.7.2, suggests that good spellings are often located far beyond the scope of small ASR n-best lists. As a result, a manual search for good spellings is infeasible, thereby strengthening the rationale for using our proposed method.

8.4.8.2 Human-in-the-loop refinement statistics

Figures 8.7 and 8.8 demonstrate that the spellings selected by our Human-in-the-Loop approach are approximately evenly distributed among the top-5 spellings generated for both ASR+HITL and ACOUSTICRANK+HITL. Notably, the superior performance of ACOUSTICRANK+HITL over ACOUSTICRANK and Figure 8.8 both suggest that our proposed acoustic ranking method is effective at reducing a large ASR n-best list to a shortlist of potentially good spellings. However, our method did not consistently rank the human-selected spelling at the very top. This could be due to a failure to detect small differences in pronunciation. It could also be related to our observation that within the top-5 spellings retrieved for ACOUSTICRANK+HITL, sometimes all of the spellings have different and

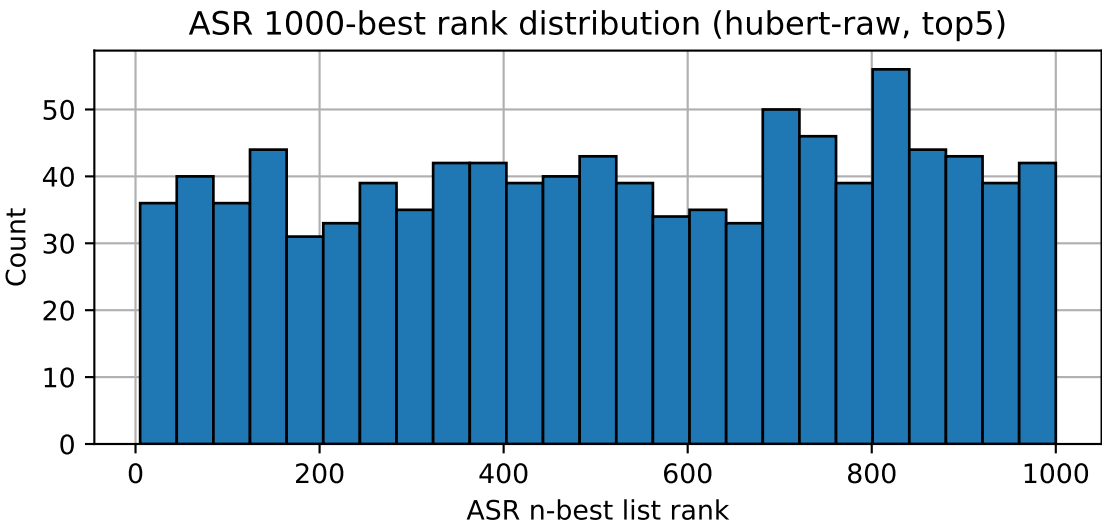

Figure 8.6: Histogram of ASR n-best ranks of the top-5 spellings retrieved using HUBERT-RAW/ACOUSTICRANK for the 200 evaluated wordtypes from Experiments 1 and 2.

slight mispronunciations, requiring the Human-in-the-Loop to select one of these spellings somewhat arbitrarily.

8.5 Discussion

In Section 8.4.6.1 we observed the relatively poor performance of HUBERT-CENTROID when compared to the strong performance of HUBERT-CODE. This was unexpected since they are conceptually similar features. One potential explanation is that collapsing speech representations down to cluster centroids potentially loses important fine-grained pronunciation information needed to discriminate between good and poor spelling candidates.

One possible explanation for the strong performance of HUBERT-CODE compared to HUBERT-CENTROID could be attributed to the potential *suitability* of using edit distance. Edit distance, which comprises additions, deletions, and substitutions to transform one sequence of symbols into another, is well suited for calculating acoustic distances between sequences of discretised speech codes. Prior work from Dieleman et al. (2021), Chorowski et al. (2021), and Yeh and Tang (2023) has observed that sequences of SSL speech representations vary in a 'fast' manner in the temporal dimension. That is, the latent representations output from a vanilla SSL model such as HuBERT tend to vary significantly from frame-to-frame,

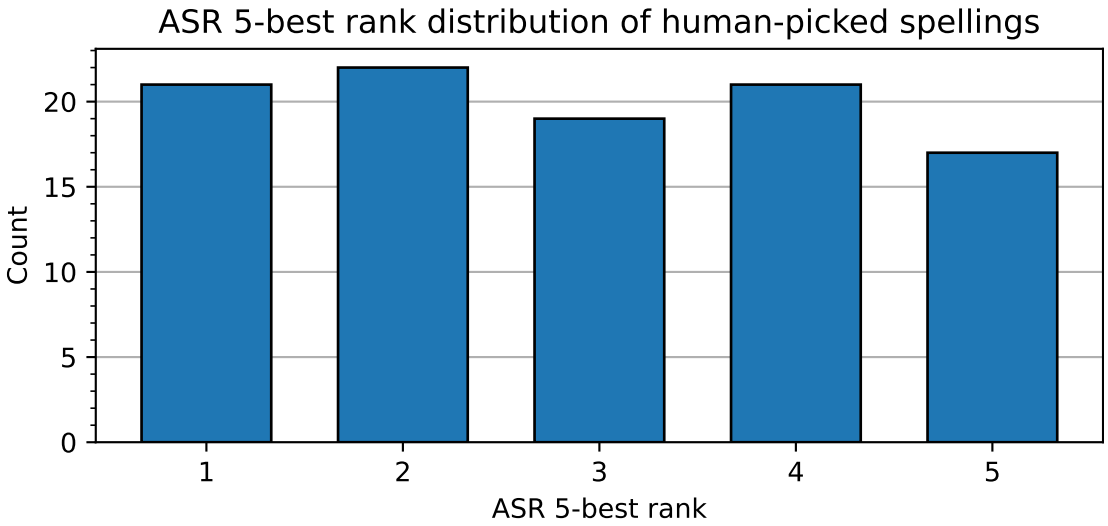

Figure 8.7: Distribution of ASR n-best ranks of the human selected spellings in ASR+HitL.

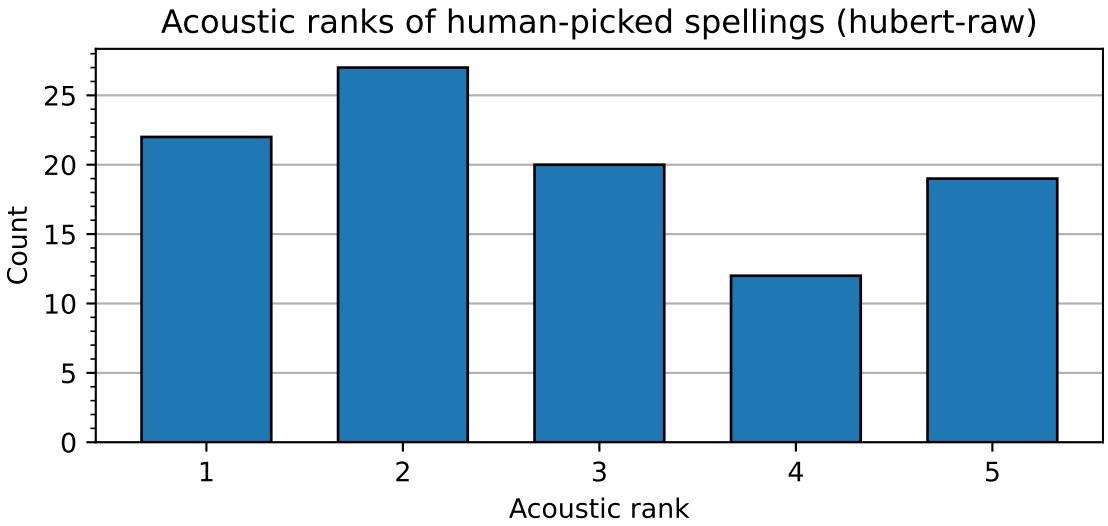

Figure 8.8: Distribution of acoustic ranks of the human selected spellings in AcousticRank+HitL.

discretising into different codes, even when the underlying phonemic category remains constant. Such fast varying discrete code sequences can exhibit long sequences of a single code interjected with another code, for example in the following sequence '2,2,2,2,**3**,2,2,2,2', a sequence of two's contains a single three, which could be regarded as 'noise'. If we consider the sequences '2,2,2,2,**3**,2,2,2,2' and '2,2,2,2,**2**,2,2,2,2' it would take only one substitution to transform one sequence into the other, resulting in a relatively small edit distance. On the other hand, the distance between the same sequences represented in HUBERT-CENTROID space could be much larger, especially if the cluster centroids of '2' and '3' are very far apart. Subsequently, it is possible that the reason why HUBERT-CODE was effective at acoustically ranking pronunciation candidates in our experiments is because edit distance is robust when comparing sequences of noisy and fast-varying speech representations. The aforementioned prior works Dieleman et al. (2021), Chorowski et al. (2021), and Yeh and Tang (2023) have sought to modify the architecture of SSL speech models in order to make their latent speech representations vary more slowly in the time dimension. According to our argument that posited why HUBERT-CODE is more effective than HUBERT-CENTROID, their performance disparity would likely narrow if using slower-varying speech representations.

Additionally in this work we use the standard dimensions used for each feature type. For example, for MFCCs we use 13 features, and for HUBERT-RAW we use 768 dimensions. Although we could have used dimensionality reduction in order to more fairly compare across different feature types, in this work we are primarily concerned with finding the feature type that works *best* for our task.

8.5.1 Future work

To enhance the Spell4TTS method, we recognise numerous opportunities for future research that involve delving deeper into its design decisions and identifying potential limitations. By exploring these avenues, we can advance our understanding and improve the overall effectiveness of the Spell4TTS approach.

Firstly, deeply understanding the acoustic distance measure would lead to greater confidence in relying on it to discover good spellings. Future work should seek to further investigate design choices such as the SSL speech representation model used, using slower or faster varying SSL representations, normalising features across utterances or speakers, and changing the number of k-means clusters used

to discretise SSL speech representations.

Additionally, future work should investigate how the phonetic coverage of the alphabet inventory might impact efficacy of our method. It is plausible that certain alphabets may not possess sufficient coverage to capture all of its speech sounds. Addressing this issue may involve incorporating additional characters or diacritics to represent unrepresented sounds, although this would necessitate training human annotators. Despite the associated costs, this approach could remain feasible if the expenses involved in this type of training are lower than those required for phonemic transcription of pronunciations.

Finally there a variety of projects that could be explored to either improve our method or further identify its potential use cases:

- Modifying Spell4TTS to extract pronunciations from multi-speaker corpora, potentially via crowdsourcing, would allow it to better suit real world production scenarios where the TTS voice talent may no longer be available.

- Incorporating stress markers and syllable boundaries may enable a more comprehensive representation of pronunciation patterns and potentially greater control when manual editing of grapheme-based pronunciations is required.

- Exploring whether Spell4TTS can be used to transcribe pronunciation variations between different regions or accents in a language.

- Enhancing the automated aspect of our method. We propose that one can finetune the ASR model to prioritise good spellings, perhaps by utilising the acoustic distance as a signal for reinforcement learning based optimisation. We also propose that iterative training of the TTS or ASR system using spellings discovered by our method could also improve the eventual performance of the downstream TTS model.

- Investigating whether Spell4TTS can be leveraged for automatic mining of *phoneme*-based pronunciations, potentially offering a valuable tool for linguistic field-based research.

8.6 Conclusion

This paper introduces a novel cost-effective method for improving the pronunciation of any existing text-to-speech (TTS) system, without requiring additional model training. The proposed method, Spell4TTS, automatically generates candidate spellings and then filters them via acoustic ranking. The resulting spellings are both human-interpretable and editable, making the method suitable for deployment in production environments. We believe that this method will be particularly beneficial for low-resource TTS. Our experiments demonstrate that the method devises pronunciations that outperform those obtained using an automatic speech recognition (ASR) baseline and the original spellings. Furthermore, the method can be augmented with human judgements to further enhance the quality of the pronunciations. We also show that ranking candidate spellings in the acoustic space of self-supervised speech representations, as opposed to traditional hand-engineered features, can yield further improvements in pronunciation.

Chapter 9

Conclusion

This thesis aimed to discover methods that can control the pronunciation of a TTS system using as few linguistic resources as possible. Through my research, I successfully developed and proposed three novel methods that meet this goal.

The first method, described in Chapter 5, involves using a compact and focused phoneme-based lexicon to train a grapheme-phoneme representation mixing TTS model. While this approach retains control over pronunciation, it necessitates a balance between the robustness of corrections and the amount of phoneme transcriptions required.

Seeking to eliminate the use of a phoneme-based lexicon, the second method, detailed in Chapter 7, centres around the use of ground-truth spoken examples, which are simpler and cheaper to obtain than phoneme transcriptions. This technique not only proved to be very effective in controlling TTS pronunciation but also allows fine-grained control over aspects such as duration and syllable stress; however, it relies on pronunciation entries which are hard to interpret and adds complexity to the TTS training process.

The third method, explored in Chapter 8, sought to generate more interpretable pronunciations from spoken examples. It involves generating candidate spellings from the hypotheses of a speech recognition model, and then using an acoustic distance to retrieve spellings that, when synthesised, are close to the ground-truth spoken example. This method was highly effective in controlling TTS pronunciation, but potentially may be somewhat limited by the phonetic expressivity of the grapheme set of the target language.

Each of these novel techniques constitute a significant contribution to the field of text-to-speech (TTS) systems, particularly for languages with limited linguistic resources. The techniques advance two previously under-explored areas of grapheme-input TTS and pronunciation control. They offer practical solutions which are accessible to non-experts. At a minimum, users require only spoken examples to represent pronunciations, eliminating the need for phoneme transcriptions. While the direct application to low-resource languages has not been explicitly demonstrated in this thesis, it is anticipated that these techniques would be effective with minimal or no modifications.

In concluding this thesis, it is worth underscoring its potential societal and cultural impact. This work is a vital step in bridging the digital divide, as it significantly advances technological inclusivity and linguistic diversity. It democratises access to high-quality TTS, empowering communities with greater control and agency over the systems that they use. Moreover, these advancements will potentially lead to further development and interest in accessible and inclusive speech technology solutions going forward.

9.1 Future work

Future research avenues emerge from the techniques presented in this thesis. Specifically, in Chapter 7, the current method for controlling TTS pronunciation using spoken examples necessitates external alignments. An advancement would be to develop a model capable of encoding both text and speech to subsequently decode speech, thereby eliminating the need for these external alignments. During inference, this model would accept an unaligned spoken example into its encoder, utilising it to make pronunciation corrections. Such a model may require a noise or masking-based training scheme, where words and segments of speech are randomly corrupted, to foster the encoder's ability to both utilise both modalities, and implicitly learn the alignment between words and their spoken forms in the input.

In Chapter 8, although the retrieval of spellings proved effective in controlling TTS pronunciation, the interpretability of these retrieved spellings was sometimes lacking. We posit that this was the case because the implicit front-end of the neural TTS system internalises complex letter-to-sound mappings, that are not intuitive to human speakers of a language, yet can be used to generate closer

acoustic matches. To address this, future research might investigate the use of a grapheme-based language model to incorporate a perplexity measure into the loss function. This integration might compel the ASR model to generate character sequences that more closely resemble conventional English spellings. Achieving this goal would involve finetuning the ASR model specifically for spelling retrieval, tailoring its output to align more with standard orthographic conventions, while still producing spellings which, when synthesised by the target TTS system, are close to ground-truth spoken examples.

Looking beyond my proposed techniques, in the fields of text and image generation, there is a clear shift towards prioritising data quantity over quality. This is most evident in the rise of large language Models (LLMs) (Brown et al., 2020; Touvron et al., 2023) and diffusion based text-to-image generation technologies (Ramesh et al., 2022; Vyas et al., 2023). Models trained on extensive datasets exhibit versatility across a diverse array of tasks. In a similar vein, there have been recent developments in speech-based generative models which train on large amounts of data. These models autoregressively generate speech and leverage multiple sources of training data to improve the naturalness and expressivity of generated speech (Rubenstein et al., 2023). Furthermore, they demonstrate novel capabilities such as text-based prompting to control speech aspects such as speaker identity, channel conditions, and speaking style. With regards to pronunciation control, future work could investigate whether these models could be adapted to retrieve the pronunciation of words in both high and low-resource languages using a small amount of labelled or paired data.

An additional avenue for future research could build upon existing advancements in speech-based self-supervised learning models. Given the relative ease of acquiring speech data without accompanying transcriptions, this untapped resource could be leveraged to source correct pronunciations. For instance, one could pretrain a TTS decoder using a large, speech-only dataset encompassing a wide range of pronunciations. This pretrained decoder could then be integrated with a TTS encoder. The training process would be designed to enable the system to map unfamiliar words during inference to their correct pronunciations (which were encountered by the decoder in the pretraining phase). This approach could enhance the model's ability to retrieve pronunciation knowledge for new, unseen words, thereby improving the overall robustness and accuracy of the TTS system.

Bibliography

Alammar, J. (2024). The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer/.

Alharbi, S., Alrazgan, M., Alrashed, A., Alnomasi, T., Almojel, R., Alharbi, R., Alharbi, S., Alturki, S., Alshehri, F., and Almojil, M. (2021). Automatic speech recognition: Systematic literature review. *IEEE Access*, 9:131858–131876.

Amazon (2019). Voices in Amazon Polly. https://docs.aws.amazon.com/polly/latest/dg/voicelist.html.

Árnason, K. (2011). *The phonology of Icelandic and Faroese*. Oxford University Press.

Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460.

Bagshaw, P. C. (1998). Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression. *Computer Speech & Language*, 12(2):119–142.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*, San Diego, USA.

Bailly, G., Lenglet, M., Perrotin, O., and Klabbers, E. (2023). Advocating for text input in multi-speaker text-to-speech systems. In *SSW 2023-12th ISCA Speech Synthesis Workshop (SSW2023)*, pages 1–7. ISCA.

Ball, R. (1999). Spelling reform in france and germany: Attitudes and reactions. *Current Issues in Language & Society*, 6(3-4):276–280.

Baluch, B. (2013). Persian orthography and its relation to literacy. In *Handbook of orthography and literacy*, pages 365–376. Routledge.

Barnes, M. P. and Weyhe, E. (2013). Faroese. In *The Germanic Languages*, pages 190–218. Routledge.

Bartelds, M., Richter, C., Liberman, M., and Wieling, M. (2020). A new acoustic-based pronunciation distance measure. *Frontiers in Artificial Intelligence*, 3:39.

Bellman, R. and Kalaba, R. (1959). On adaptive control processes. *IRE Transactions on Automatic Control*, 4(2):1–9.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bergmann, J. and Wimmer, H. (2008). A dual-route perspective on poor reading in a regular orthography: Evidence from phonological and orthographic lexical decisions. *Cognitive Neuropsychology*, 25(5):653–676.

Beyer, S. V. (1992). *The classical Tibetan language*. SUNY Press.

Black, A. and Lenzo, K. (2000). Building voices in the festival speech synthesis system.

Black, A. W. and Taylor, P. A. (1997). Automatically clustering similar units for unit selection in speech synthesis.

Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Braunschweiler, N., Gales, M. J., and Buchholz, S. (2010). Lightly supervised recognition for automatic alignment of large coherent speech recordings. In *Proc. Interspeech*, Makuhari, Chiba, Japan.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Campbell, L. (2013). *Historical linguistics*. Edinburgh University Press.

Chorowski, J., Ciesielski, G., Dzikowski, J., Łańcucki, A., Marxer, R., Opala, M.,

Pusz, P., Rychlikowski, P., and Stypułkowski, M. (2021). Aligned Contrastive Predictive Coding. In *Proc. Interspeech 2021*, pages 976–980.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Chung, Y.-A. and Glass, J. (2020). Generative pre-training for speech with autoregressive predictive coding. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3497–3501. IEEE.

Chung, Y.-A., Wang, Y., Hsu, W.-N., Zhang, Y., and Skerry-Ryan, R. (2019). Semi-supervised training for improving data efficiency in end-to-end speech synthesis. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom.

Clark, R. A., Richmond, K., and King, S. (2007a). Multisyn: Open-domain unit selection for the Festival speech synthesis system. *Speech Communication*, 49(4):317–330.

Clark, R. A. J., Richmond, K., and King, S. (2007b). Multisyn: Open-domain unit selection for the Festival speech synthesis system. *Speech Communication*, 49(4):317–330.

CMU (2019). The Carnegie Mellon pronouncing dictionary. https://github.com/cmusphinx/cmudict.

CMUdict (2014). Carnegie mellon pronouncing dictionary.

Coltheart, M. (2005). Modeling reading: The dual-route approach. *The science of reading: A handbook*, pages 6–23.

CSTR (2011). The Nancy corpus. http://www.cstr.ed.ac.uk/projects/blizzard/2011/lessac_blizzard20\11/.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Dieleman, S., Nash, C., Engel, J., and Simonyan, K. (2021). Variable-rate discrete representation learning. *arXiv preprint arXiv:2103.06089*.

Dunbar, E., Algayres, R., Karadayi, J., Bernard, M., Benjumea, J., Cao, X.-N., Miskic, L., Dugrain, C., Ondel, L., Black, A. W., et al. (2019). The zero resource speech challenge 2019: TTS without T. *arXiv preprint arXiv:1904.11469*.

Ebden, P. and Sproat, R. (2015). The Kestrel TTS text normalization system. *Natural Language Engineering*, 21(3):333—-353.

Elovitz, H., Johnson, R., McHugh, A., and Shore, J. (1976). to-sound rules for automatic translation of english text to phonetics. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(6):446–459.

Ethnologue (2019). How many languages are there in the world? https://www.ethnologue.com/guides/\how-many-languages.

Fitt, S. (2001). Unisyn lexicon release (version 1.3)[datafile and codebook]. *Edinburgh, Scotland: Centre for Speech Technology Research at the University of Edinburgh*.

Fitt, S. (2018). Unisyn Lexicon. http://www.cstr.ed.ac.uk/projects/unisyn/.

Fitt, S. and Richmond, K. (2006). Redundancy and productivity in the speech technology lexicon - can we do better? In *Proc. Interspeech*, pages 1202–1204.

Fong, J., Lyth, D., Henter, G. E., Tang, H., and King, S. (2022a). Speech audio corrector: using speech from non-target speakers for one-off correction of mispronunciations in grapheme-input text-to-speech. *Proc. Interspeech 2022*, pages 1213–1217.

Fong, J., Tang, H., and King, S. (2023). Spell4TTS: Acoustically-informed spellings for improving text-to-speech pronunciations. In *12th Speech Synthesis Workshop (SSW) 2023*.

Fong, J., Taylor, J., and King, S. (2020). Testing the limits of representation mixing for pronunciation correction in end-to-end speech synthesis. In *INTERSPEECH*, pages 4019–4023.

Fong, J., Taylor, J., Richmond, K., and King, S. (2019a). A comparison between letters and phones as input to sequence-to-sequence models for speech synthesis. In *10th ISCA Speech Synthesis Workshop*.

Fong, J., Taylor, J., Richmond, K., and King, S. (2019b). Investigating the robust-

ness of sequence-to-sequence text-to-speech models to imperfectly-transcribed training data. In *Interspeech 2019*.

Fong, J., Wang, Y., Agrawal, P., Manohar, V., Wu, J., Köhler, T., and He, Q. (2022b). Towards zero-shot text-based voice editing using acoustic context conditioning, utterance embeddings, and reference encoders. *arXiv preprint arXiv:2210.16045*.

Fong, J., Williams, J., and King, S. Analysing Temporal Sensitivity of VQ-VAE Sub-Phone Codebooks.

Fong, J., Williams, J., and King, S. (2021). Analysing temporal sensitivity of vq-vae sub-phone codebooks. In *The 11th ISCA Speech Synthesis Workshop (SSW11)*, pages 27–231.

Fong, J., Wu, J., Agrawal, P., Gibiansky, A., Koehler, T., and He, Q. Improving polyglot speech synthesis through multi-task and adversarial learning.

Geerts, G., Van Den Broeck, J., and Verdoodt, A. (1977). Successes and failures in Dutch spelling reform. *Advances in the creation and revision of writing systems*, pages 179–245.

Goodman, K. S. (1993). *Phonics phacts*. ERIC.

Google (2019). Cloud Text-to-Speech. https://cloud.google.com/text-to-speech/.

GORDOS, G., OLASZY, G., and SABAH, M. (1992). Acoustic building units for formant synthesis text-to-speech converter system for modern standard arabic. *Periodica Polytechnica Electrical Engineering (Archives)*, 36(1):39–52.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Griffin, D. and Lim, J. (1984). Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243.

Hayashi, T. and Watanabe, S. (2020). Discretalk: Text-to-speech as a machine translation problem. *arXiv preprint arXiv:2005.05525*.

Henter, G. E., Lorenzo-Trueba, J., Wang, X., and Yamagishi, J. (2018). Deep encoder-decoder models for unsupervised learning of controllable speech synthesis. *arXiv preprint arXiv:1807.11470*.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hodges, R. E. (1964). A short history of spelling reform in the united states. *The Phi Delta Kappan*, 45(7):330–332.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70.

Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., and Mohamed, A. (2021). HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. *arXiv preprint arXiv:2106.07447*.

Hsu, W.-N., Zhang, Y., and Glass, J. (2017). Unsupervised learning of disentangled and interpretable representations from sequential data. *arXiv preprint arXiv:1709.07902*.

Huang, R., Zhang, C., Wang, Y., Yang, D., Liu, L., Ye, Z., Jiang, Z., Weng, C., Zhao, Z., and Yu, D. (2023). Make-a-voice: Unified voice synthesis with discrete representation. *arXiv preprint arXiv:2305.19269*.

Hunt, A. J. and Black, A. W. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 373–376. IEEE.

Ito, K. and Johnson, L. (2017). The lj speech dataset. https://keithito.com/LJ-Speech-Dataset/.

ITU (2002). Method for subjective assessment of intermediate quality level of coding systems. Technical Report ITU-R BS.1534-1, International Telecommunication Union.

Jackson, N. E. and Coltheart, M. (2013). *Routes to reading success and failure: Toward an integrated cognitive psychology of atypical reading*. Psychology Press.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jia, Y., Zen, H., Shen, J., Zhang, Y., and Wu, Y. (2021). PnG BERT: Augmented BERT on Phonemes and Graphemes for Neural TTS. In *Proc. Interspeech 2021*, pages 151–155.

Jia, Y., Zhang, Y., Weiss, R., Wang, Q., Shen, J., Ren, F., Nguyen, P., Pang, R., Moreno, I. L., Wu, Y., et al. (2018). Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4485–4495, Montréal, Canada.

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., Oord, A., Dieleman, S., and Kavukcuoglu, K. (2018). Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419. PMLR.

Kanters, S., Cucchiarini, C., and Strik, H. (2009). The goodness of pronunciation algorithm: a detailed performance study.

Kastner, K., Santos, J. F., Bengio, Y., and Courville, A. (2019). Representation mixing for tts synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5906–5910. IEEE.

Katz, L. and Frost, R. (1992). The reading process is different for different orthographies: The orthographic depth hypothesis. In *Advances in psychology*, volume 94, pages 67–84. Elsevier.

Kawahara, H., Masuda-Katsuse, I., and De Cheveigne, A. (1999). Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds. *Speech communication*, 27(3-4):187–207.

Kharitonov, E., Copet, J., Lakhotia, K., Nguyen, T. A., Tomasello, P., Lee, A., Elkahky, A., Hsu, W.-N., Mohamed, A., Dupoux, E., et al. (2022). textless-lib: a Library for Textless Spoken Language Processing. *arXiv preprint arXiv:2202.07359*.

Kharitonov, E., Lee, A., Polyak, A., Adi, Y., Copet, J., Lakhotia, K., Nguyen, T.-A., Rivière, M., Mohamed, A., Dupoux, E., et al. (2021). Text-free prosody-aware generative spoken language modeling. *arXiv preprint arXiv:2109.03264*.

Kim, J., Kim, S., Kong, J., and 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.

Kim, M., Jeong, M., Choi, B. J., Ahn, S., Lee, J. Y., and Kim, N. S. (2022). Transfer learning framework for low-resource text-to-speech using a large-scale unlabeled speech corpus. *arXiv preprint arXiv:2203.15447*.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proc. ICLR*, San Diego, USA.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-source toolkit for neural machine translation. In *ACL 2017 - Annual Meeting of the Association for Computational Linguistics, Proceedings*, pages 67–72.

Kong, J., Kim, J., and Bae, J. (2020). HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in Neural Information Processing Systems*, 33:17022–17033.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kuhl, P. K. (1991). Human adults and human infants show a "perceptual magnet effect" for the prototypes of speech categories, monkeys do not. *Perception & psychophysics*, 50(2):93–107.

Lakhotia, K., Kharitonov, E., Hsu, W.-N., Adi, Y., Polyak, A., Bolte, B., Nguyen, T.-A., Copet, J., Baevski, A., Mohamed, A., et al. (2021). On generative spoken language modeling from raw audio. *Transactions of the Association for Computational Linguistics*, 9:1336–1354.

Łańcucki, A. (2021). Fastpitch: Parallel text-to-speech with pitch prediction. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6588–6592. IEEE.

Latorre, J., Lachowicz, J., Lorenzo-Trueba, J., Merritt, T., and Drugman, T. (2019a). Effect of data reduction on sequence-to-sequence neural TTS. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom.

Latorre, J., Lachowicz, J., Lorenzo-Trueba, J., Merritt, T., Drugman, T., Ronanki, S., and Klimkov, V. (2019b). Effect of data reduction on sequence-to-sequence neural tts. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7075–7079. IEEE.

Lee, A., Chen, P.-J., Wang, C., Gu, J., Ma, X., Polyak, A., Adi, Y., He, Q., Tang, Y., Pino, J., et al. (2021a). Direct speech-to-speech translation with discrete units. *arXiv preprint arXiv:2107.05604*.

Lee, A., Gong, H., Duquenne, P.-A., Schwenk, H., Chen, P.-J., Wang, C., Popuri, S., Pino, J., Gu, J., and Hsu, W.-N. (2021b). Textless Speech-to-Speech Translation on Real Data. *arXiv preprint arXiv:2112.08352*.

Li, N., Liu, S., Liu, Y., Zhao, S., and Liu, M. (2019). Neural speech synthesis with transformer network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6706–6713.

Li, Y., Mohamied, Y., Bell, P., and Lai, C. (2023). Exploration of a self-supervised speech model: A study on emotional corpora. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 868–875. IEEE.

Liberman, I. Y., Shankweiler, D., and Liberman, A. M. (1989). The alphabetic principle and learning to read.

Lindblom, B. (1990). Explaining phonetic variation: A sketch of the h&h theory. In *Speech production and speech modelling*, pages 403–439. Springer.

Luong, M.-T., Pham, H., and Manning, C. D. (2015a). Effective approaches to attention-based neural machine translation. In *Proc. ACL*, Beijing, China.

Luong, M.-T., Pham, H., and Manning, C. D. (2015b). Effective approaches to attention-based neural machine translation. In *ACL - Annual Meeting of the Association for Computational Linguistics, Proceedings*, pages 1412–1421.

Ma, J. (2024). All of Recurrent Neural Networks. https://medium.com/@jianqiangma/all-about-recurrent-neural-networks-9e5ae2936f6e.

Mallick, S. (2024). Understanding Convolutional Neural Network (CNN): A Complete GuideEnglish-language Alexa voice learns to speak Spanish.

Marjou, X. (2019). OTEANN: Estimating the transparency of orthographies with an artificial neural network. *arXiv preprint arXiv:1912.13321*.

McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., and Sonderegger, M. (2017). Montreal forced aligner: Trainable text-speech alignment using kaldi.

McCarthy, O. (2019). Tacotron and WaveRNN implementation. https://github.com/fatchord/WaveRNN.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

Nayernia, L., van de Vijver, R., and Indefrey, P. (2019). The influence of orthography on phonemic knowledge: An experimental investigation on german and persian. *Journal of psycholinguistic research*, 48:1391–1406.

Nguyen, T. A., de Seyssel, M., Rozé, P., Rivière, M., Kharitonov, E., Baevski, A., Dunbar, E., and Dupoux, E. (2020). The zero resource speech benchmark 2021: Metrics and baselines for unsupervised spoken language modeling. *arXiv preprint arXiv:2011.11588*.

Ning, Y., He, S., Wu, Z., Xing, C., and Zhang, L.-J. (2019). A review of deep learning based speech synthesis. *Applied Sciences*, 9(19):4050.

Novak, J. R., Minematsu, N., and Hirose, K. (2016). Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the wfst framework. *Natural Language Engineering*, 22(6):907–938.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: an ASR corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE.

Pereira, R. (2019). Linguistic Map of Orthographic depth.

https://linguisticmaps.tumblr.com/post/187856489343/orthographic-depth-languages-have-different-levels.

Perquin, A., Cooper, E., and Yamagishi, J. (2020). An investigation of the relation between grapheme embeddings and pronunciation for tacotron-based systems. *arXiv preprint arXiv:2010.10694*.

Ping, W., Peng, K., Gibiansky, A., Arik, S. O., Kannan, A., Narang, S., Raiman, J., and Miller, J. (2018). Deep Voice 3: 2000-speaker neural text-to-speech. In *Proc. ICLR*, Vancouver, Canada.

Polyak, A., Adi, Y., Copet, J., Kharitonov, E., Lakhotia, K., Hsu, W.-N., Mohamed, A., and Dupoux, E. (2021). Speech resynthesis from discrete disentangled self-supervised representations. *arXiv preprint arXiv:2104.00355*.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al. (2011). The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number CONF. IEEE Signal Processing Society.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3.

Rao, K., Peng, F., Sak, H., and Beaufays, F. (2015). Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4225–4229. IEEE.

Ren, Y., Hu, C., Tan, X., Qin, T., Zhao, S., Zhao, Z., and Liu, T.-Y. (2020). Fastspeech 2: Fast and high-quality end-to-end text to speech. *arXiv preprint arXiv:2006.04558*.

Rey, A., Ziegler, J. C., and Jacobs, A. M. (2000). Graphemes are perceptual reading units. *Cognition*, 75(1):B1–B12.

Richmond, K. (2018). Combilex speech technology lexicon. http://homepages.inf.ed.ac.uk/korin/sitenew/\Research/Combilex.

Richmond, K., Clark, R. A. J., and Fitt, S. (2009). Robust LTS rules with

the Combilex speech technology lexicon. In *Interspeech, Proceedings*, pages 1295–1298.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rosenfeld, R. and Sherwani, J. (2004). The case for speech technology for developing regions.

Rubenstein, P. K., Asawaroengchai, C., Nguyen, D. D., Bapna, A., Borsos, Z., Quitry, F. d. C., Chen, P., Badawy, D. E., Han, W., Kharitonov, E., et al. (2023). AudioPaLM: A Large Language Model That Can Speak and Listen. *arXiv preprint arXiv:2306.12925*.

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

Schröder, M. and Trouvain, J. (2003). The German text-to-speech synthesis system MARY: A tool for research, development and teaching. *International Journal of Speech Technology*, 6(4):365–377.

Shen, J. et al. (2018a). Natural TTS synthesis by conditioning WaveNet on Mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Proceedings*, pages 4779–4783.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., et al. (2018b). Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783. IEEE.

Sheoran, K., Bajgoti, A., Gupta, R., Jatana, N., Dhand, G., Gupta, C., Dadheech, P., Yahya, U., and Aneja, N. (2023). Pronunciation scoring with goodness of pronunciation and dynamic time warping. *IEEE Access*, 11:15485–15495.

Sicherman, A. and Adi, Y. (2023). Analysing discrete self supervised speech representation for spoken language modeling. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Slowiaczek, M. L. and Clifton Jr, C. (1980). Subvocalization and reading for meaning. *Journal of verbal learning and verbal behavior*, 19(5):573–582.

Sotelo, J., Mehri, S., Kumar, K., Santos, J. F., Kastner, K., Courville, A., and Bengio, Y. (2017). Char2Wav: End-to-end speech synthesis. In *Proc. ICLR*, Toulon, France.

Stan, A., Bell, P., Yamagishi, J., and King, S. (2013). Lightly supervised discriminative training of grapheme models for improved sentence-level alignment of speech and text data. In *Proc. Interspeech*, pages 1525–1529, Lyon, France.

Stuart-Smith, J. and Timmins, C. (2006). 'Tell her to shut her moof: the role of the lexicon in TH-fronting in Glaswegian. In *The power of words*, pages 171–183. Brill.

Sun, G., Zhang, Y., Weiss, R. J., Cao, Y., Zen, H., Rosenberg, A., Ramabhadran, B., and Wu, Y. (2020). Generating diverse and natural text-to-speech samples using a quantized fine-grained VAE and autoregressive prosody prior. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6699–6703. IEEE.

Sun, S., Richmond, K., and Tang, H. (2023). Improving Seq2Seq TTS Frontends with Transcribed Speech Audio. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Swadesh, M. (1934). The phonemic principle. *Language*, 10(2):117–129.

Tachibana, H., Uenoyama, K., and Aihara, S. (2018a). Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4784–4788. IEEE.

Tachibana, H., Uenoyama, K., and Aihara, S. (2018b). Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4784–4788, Calgary, Canada.

Tachibana, H., Uenoyama, K., and Aihara, S. (2018c). Efficiently trainable Text-to-Speech system based on deep convolutional networks with guided attention.

In *ICASSP - IEEE International Conference on Acoustics, Speech and Signal Processing, Proceedings*, pages 4784–4788.

Taylor, J. and Richmond, K. (2019a). Analysis of pronunciation learning in end-to-end speech synthesis. In *To appear in Interspeech 2019*.

Taylor, J. and Richmond, K. (2019b). Analysis of pronunciation learning in end-to-end speech synthesis. In *20th Annual Conference of the International Speech Communication Association: Crossroads of Speech and Language*, pages 2070–2074. International Speech Communication Association.

Taylor, P. (2009). *Text-to-speech synthesis*. Cambridge university press.

Tjandra, A., Sakti, S., and Nakamura, S. (2020). Transformer VQ-VAE for Unsupervised Unit Discovery and Speech Synthesis: ZeroSpeech 2020 Challenge.

Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., and Kitamura, T. (2000). Speech parameter generation algorithms for hmm-based speech synthesis. In *2000 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 00CH37100)*, volume 3, pages 1315–1318. IEEE.

Tokuda, K., Zen, H., and Black, A. W. (2002). An hmm-based speech synthesis system applied to english. In *IEEE speech synthesis workshop*, pages 227–230. Citeseer.

Tournadre, N. (2014). The tibetic languages and their classification. *Trans-Himalayan linguistics: Historical and descriptive linguistics of the Himalayan area*, 266(1):105–29.

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. *arXiv preprint arXiv:2302.13971*.

Tu, T., Chen, Y.-J., Yeh, C.-c., and Lee, H.-Y. (2019). End-to-end text-to-speech for low-resource languages by cross-lingual transfer learning. *arXiv preprint arXiv:1904.06508*.

Twaddell, W. F. (1935). On defining the phoneme. *Language*, 11(1):5–62.

Vakil, A., Paulus, M., Palmer, A., and Regneri, M. (2014). lex4all: A language-independent tool for building and evaluating pronunciation lexicons for small-vocabulary speech recognition. In *Proceedings of 52nd Annual Meeting of*

the Association for Computational Linguistics: System Demonstrations, pages 109–114.

Van Den Bosch, A. and Daelemans, W. (1993). Data-oriented methods for grapheme-to-phoneme conversion. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics*.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). WaveNet: A generative model for raw audio. *arXiv unreviewed manuscript arXiv:1609.03499*.

van den Oord, A., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. (2017). Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*.

van Esch, D., Chua, M., and Rao, K. (2016). Predicting pronunciations with syllabification and stress with recurrent neural networks. In *INTERSPEECH*, pages 2841–2845.

van Niekerk, B., Carbonneau, M.-A., Zaïdi, J., Baas, M., Seuté, H., and Kamper, H. (2022). A comparison of discrete and soft speech units for improved voice conversion. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6562–6566. IEEE.

van Niekerk, B., Nortje, L., Baas, M., and Kamper, H. (2021). Analyzing speaker information in self-supervised models to improve zero-resource speech processing. *arXiv preprint arXiv:2108.00917*.

van Niekerk, B., Nortje, L., and Kamper, H. (2020). Vector-quantized neural networks for acoustic unit discovery in the zerospeech 2020 challenge. *arXiv preprint arXiv:2005.09409*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, Long Beach, USA.

Veaux, C., Yamagishi, J., MacDonald, K., et al. (2017). CSTR VCTK corpus:

English multi-speaker corpus for CSTR voice cloning toolkit. *University of Edinburgh. The Centre for Speech Technology Research (CSTR).*

Vibulpatanavong, K. and Evans, D. (2019). Phonological awareness and reading in thai children. *Reading and Writing*, 32:467–491.

Vyas, A., Shi, B., Le, M., Tjandra, A., Wu, Y.-C., Guo, B., Zhang, J., Zhang, X., Adkins, R., Ngan, W., et al. (2023). Audiobox: Unified audio generation with natural language prompts. *arXiv preprint arXiv:2312.15821*.

Walker, J. (1830). *A critical pronouncing dictionary..* stereotyped by A. Wilson, for T. Cadell.

Wang, D., Zeng, M., Zhao, H., Gao, L., Li, S., Niu, Z., Bai, X., and Gao, X. (2023). Effects of syllable boundaries in tibetan reading. *Scientific Reports*, 13(1):314.

Wang, Y. et al. (2017a). Tacotron: Towards end-to-end speech synthesis. In *Interspeech, Proceedings*, pages 4006–4010.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., et al. (2017b). Tacotron: Towards end-to-end speech synthesis. In *Proc. Interspeech*, pages 4006–4010, Stockholm, Sweden.

Watts, O. (2019). Ophelia: A modified version of Kyubyong Park's DC-TTS repository, which implements a variant of the system described in Efficiently Trainable Text-to-Speech System Based on Deep Convolutional Networks with Guided Attention.

Watts, O., Henter, G. E., Fong, J., and Valentini-Botinhao, C. (2019). Where do the improvements come from in sequence-to-sequence neural tts? In *2019 ISCA Speech Synthesis Workshop (SSW)*, volume 10, pages 217–222.

Weide, R. L. (1998). The CMU pronouncing dictionary. *URL: http://www. speech. cs. cmu. edu/cgibin/cmudict*.

Wells, D., Tang, H., and Richmond, K. (2022). Phonetic analysis of self-supervised representations of english speech. In *23rd Annual Conference of the International Speech Communication Association, INTERSPEECH 2022*, pages 3583–3587. ISCA.

Williams, J., Fong, J., Cooper, E., and Yamagishi, J. (2021). Exploring

disentanglement with multilingual and monolingual vq-vae. *arXiv preprint arXiv:2105.01573*.

Winskel, H. and Iemwanthong, K. (2010). Reading and spelling acquisition in thai children. *Reading and Writing*, 23:1021–1053.

Witt, S. M. (2000). *Use of speech recognition in computer-assisted language learning*. PhD thesis.

Wu, Z., Watts, O., and King, S. (2016). Merlin: An open source neural network speech synthesis system. In *9th ISCA Speech Synthesis Workshop, Proceedings*.

Yamagishi, J., Ling, Z., and King, S. (2008). Robustness of HMM-based speech synthesis. In *Proc. Interspeech*, Brisbane, Australia.

Yamagishi, J., Veaux, C., MacDonald, K., et al. (2019). CSTR VCTK Corpus: English Multi-Speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92).

Yang, S.-w., Chi, P.-H., Chuang, Y.-S., Lai, C.-I. J., Lakhotia, K., Lin, Y. Y., Liu, A. T., Shi, J., Chang, X., Lin, G.-T., et al. (2021). SUPERB: Speech processing Universal PERformance Benchmark. *arXiv preprint arXiv:2105.01051*.

Yasuda, Y., Wang, X., and Yamagishd, J. (2021a). End-to-end text-to-speech using latent duration based on vq-vae. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5694–5698. IEEE.

Yasuda, Y., Wang, X., and Yamagishi, J. (2021b). Investigation of learning abilities on linguistic features in sequence-to-sequence text-to-speech synthesis. *Computer Speech & Language*, 67:101183.

Yeh, S.-L. and Tang, H. (2023). Learning dependencies of discrete speech representations with neural hidden markov models. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Yolchuyeva, S., Németh, G., and Gyires-Tóth, B. (2020). Transformer based grapheme-to-phoneme conversion. *arXiv preprint arXiv:2004.06338*.

Yoshimura, T., Tokuda, K., Masuko, T., Kobayashi, T., and Kitamura, T. (1999). Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis. In *Sixth European conference on speech communication and technology*.

Zachrisson, R. E. (1931). Four hundred years of english spelling reform. *Studia neophilologica*, 4(1):1–69.

Zen, H. (2006). An example of context-dependent label format for HMM-based speech synthesis in English.

Zen, H. et al. (2019). LibriTTS: A corpus derived from LibriSpeech for Text-to-Speech. In *Submission to Interspeech*.

Zen, H., Nose, T., Yamagishi, J., Sako, S., Masuko, T., Black, A. W., and Tokuda, K. (2007). The hmm-based speech synthesis system (hts) version 2.0. *SSW*, 6:294–299.

Ziegler, J. C. and Goswami, U. (2005). Reading acquisition, developmental dyslexia, and skilled reading across languages: a psycholinguistic grain size theory. *Psychological bulletin*, 131(1):3.