THE UNIVERSITY
of EDINBURGH

Using Local Linguistic Context for Text-to-Speech

Pilar Oplustil-Gallegos

Doctor of Philosophy
University of Edinburgh
2022

Declaration

I hereby declare that this thesis was composed by myself and that the work in this thesis is my own, except where explicitly stated otherwise in the text. In such cases where the work was completed in collaboration, I have made substantial contributions to the work and the contributions are clearly indicated. The work in this thesis has not been submitted for any other degree or professional qualification.

Pilar Oplustil Gallegos

Abstract

Synthetic speech generated by state-of-the-art Text-to-Speech (TTS) models achieves unprecedented levels of naturalness. Training, inference and evaluation of TTS models has consistently been performed on isolated utterances stripped of contextual information, despite evidence from linguistics that context can affect speech. In this thesis, we hypothesize that we can further improve synthetic speech naturalness by leveraging local linguistic context, which we define as the utterance that immediately precedes another one, considering both its textual and acoustic contents, with a focus on the latter.

The experimental work on this thesis is divided into three parts. In the first part, we develop and test a method to condition sequence-to-sequence TTS models on representations of the context utterance. Preliminary results conditioning on an acoustic representation show that it is possible to improve synthetic speech with our method, when evaluating single utterances through listening tests. Next, we systematically compare different context representations, and we find significantly better naturalness scores when combining acoustic and textual representations from context to condition TTS systems.

In the second part, we explore alternative methods to incorporate contextual information. We do not find improvements by conditioning inference only on context representations, or by augmenting the TTS input with features extracted from textual context.

In the last part of this thesis we analyse and evaluate the best method proposed in part one. We begin by testing our method on several challenging data sets of diverse nature, establishing its limitations. Subsequently, we evaluate our method by applying an in-context listening test design proposed in previous work. Unexpectedly, we see that ground-truth speech might not be considered more natural when listened to in-context than as isolated utterances, contrary to previous results.

We finish by proposing to apply local coherence models, trained on sequences of natural speech data, as an objective evaluation of synthetic speech in-context. Through this evaluation, we see that our method, using ground-truth acoustic context, provides improvements in-context, only when trained with speech from a speaker with high predictability at the local linguistic context level, encoded through acoustic features alone.

Acknowledgements

I would like to extend my sincere thanks to my supervisor Simon King, without whom this thesis would not have been possible, for his continuous support throughout my PhD and his thorough revisions of earlier drafts. I would also like to thank my second supervisor, Catherine Lai, and all the people I have had the chance to collaborate with during this PhD, for their excellent work and for their helpful contributions to my thesis.

During this PhD I had the opportunity to work for the former Sonantic and to intern for the UK Alexa TTS team of Amazon, both of which gave me much knowledge and experience of importance for my career. I would like to thank all members I worked with from both of these teams for giving me the chance to enrich my PhD through these experiences.

I'm extremely grateful to my family who, despite being at a great distance, have supported me throughout my PhD. I would not have been able to finish this PhD without the unconditional support of my partner Matthew, to whom I will always be grateful, not to mention the further encouragement I always received from his wonderful family. I want to give special thanks to my friend Francesca for her moral support and companionship.

Lastly, I would like to acknowledge professor Domingo Román, who was not just the first person to teach me about phonetics and speech, but rather the one who instilled in me a passion for the speech sciences. I will always be grateful as I would never have taken this path without his unwavering belief in me.

My PhD was funded by ANID, Becas Chile, no 72190135.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Text-to-Speech (TTS) systems transform text into a speech waveform, with the aim of generating synthetic speech as natural as human speech. State-of-the-art TTS systems are trained on a data set through computational machine learning methods to predict speech from text. TTS systems are widely used for applications such as digital assistants and dialogue systems, audiobooks, automatic dubbing, transportation announcements (in train stations or airports), among others. Although state-of-the-art TTS systems can generate synthetic speech with unprecedented levels of naturalness, synthetic voices still lack the nuanced phonetic and prosodic subtleties that are essential to natural speech in human communication.

One of the fundamental problems of generating synthetic speech from text alone is that text does not provide all the necessary information to fully predict natural speech (Taylor & Black, 1999). Among the many linguistic and paralinguistic elements that can influence a speaker's rendition of a particular text, there is context. However, context has been consistently ignored in TTS, even though it has been shown that context influences speech, especially, in terms of prosody (Section 1.2). We believe that by leveraging context within TTS systems could result in improved synthetic speech naturalness, as the TTS system is provided with more information than the text in order to predict newly generated speech. However, context is a broad and ambiguous concept with multiple dimensions, some of which are easier to formalize computationally than others.

Therefore, first, we start this thesis in Section 1.1 by discussing how to define context from a linguistic perspective, in order to identify a type of context that we can formalize computationally for improving TTS. Then, in Section 1.2 we review studies from phonetics that describe different ways in which speech is affected by context. These studies provide evidence that serves as the primary motivation for our research. Then, in Section 1.3 we give an overview of the research carried out in this thesis, together with a summary of its main contributions.

1.1 Local linguistic context

Context is a broad concept. It is common to use it in a daily conversation to convey "the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood." (*Oxford English dictionary*, 2000). Context as an object of study in linguistics has been classically associated with pragmatics in contrast to semantics: while semantics is the study of meaning,

pragmatics is the study of meaning in context. However, in practice, there is no unanimous definition of context (Finkbeiner, Meibauer, & Schumacher, 2012).

From a theoretical perspective, we can identify two different approaches when considering context in the pragmatics literature. On one hand, a group of theories focus on context as the external, social, and cognitive aspects of the communicative situation (Van Dijk, 1999). On the other hand, other approaches focus on context in discourse as a sequence of related utterances. An utterance[1] can be defined as a "stretch of speech preceded and followed by silence or a change of speaker" (Crystal, 2011). Among the latter, Roberts (2006) highlights how context influences the semantic interpretation of meaning in two fundamental ways: the aptness or felicity of an utterance depends on its context, and the context is influenced by every successive utterance in a discourse. Crucially, this means that an utterance cannot be interpreted in isolation, but rather always in context, and that there is an interdependence between every utterance and its context.

Some models of context incorporate both theoretical perspectives. Finkbeiner et al. (2012) describes the classical model of context for text, summarised in Table 1.1. In comparison, Bunt's (2000) model, described in Table 1.2, is for dialogue, and is oriented towards computational pragmatics. In both cases, we can consider that each dimension in a lower row of each table encompasses all the previous ones, as illustrated in Figure 1.1.

Context dimension	**Description**
Intra-textual (co-text)	The relation of a piece of text to its surrounding text.
Infra-textual	The relation of a piece of text to the whole of the text.
Inter-textual	The relation of a text to other texts.
Extra-textual (situational or communicative)	The relation of a text to aspects of the situation in which the text has been produced or interpreted.

Table 1.1: A model of context for text (Finkbeiner et al., 2012).

Considering these models, we can start to limit the scope of what we can work with as context in this thesis. It can be seen that the broader the context dimension, the harder it seems to be able to model it computationally. For example, there are no means to model participant mental states with current machine learning methods. Nevertheless, the first dimension of both context models, which are very similar, seems possible to formalize computationally with current technology, as it considers that context is the linguistic material that surrounds a particular textual or discursive unit, and the relationship between them.

For each context dimension, Bunt (1994) distinguishes between global and local context. Global elements are constant through out the interaction, while local elements unfold through the discourse. For example, when proposing an implementation for automatic agent dialogues, Bunt (2000) suggests that an agent should keep representations of the utterances, both from the user and from the agent in

[1]In contrast, in this thesis we refer to sentences when considering text only.

Context dimension	**Description**
Linguistic	Surrounding linguistic material, "raw" as well as analysed.
Semantic	State of the underlying task; facts in the task domain.
Cognitive	Participants' states of processing and models of each other's states.
Physical and perceptual	Availability of communicative and perceptual channels; partners' presence and attention.
Social	Communicative rights, obligations and constraints of each participant.

Table 1.2: A model of context for dialogue (Bunt, 2000). An underlying task is what motivates dialogue for each participant.

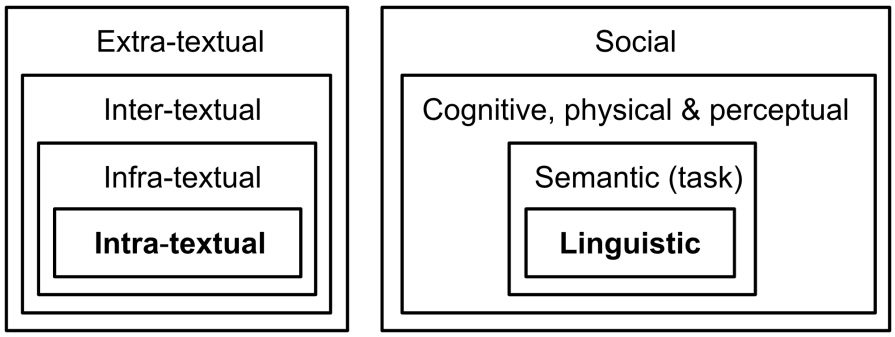

Figure 1.1: Diagram of the models of context for text on the left (Finkbeiner et al., 2012), and dialogue on the right (Bunt, 2000).

a feed-back fashion. This is denominated local linguistic context, and it is seen as a memory of the dialogue. Some advantages of limiting the scope of context to only a local one is to restrict the actual span of linguistic material to encode. An approximation of the global context of a discourse can be achieved by considering all local contexts. For example, this approach has been taken in Natural Language Processing (NLP) in the analysis of textual coherence, where the coherence of a whole document is approximated by the average local coherence of every pair of sentences (Jurafsky & Martin, 2021, Chapter 22).

In this thesis we propose to improve synthetic speech generated with TTS systems by conditioning on context, where we define and limit context as **linguistic** and **local**. Local linguistic context is possible to model with existing machine learning methods, and it is a concept that can be applied to both written and spoken discourse. Because TTS is based on modelling utterances, this will be our fundamental linguistic unit. As such, every time we refer to context in this thesis, unless stated otherwise, we refer to the linguistic material surrounding a specific utterance, both in its written and spoken form, and considering both raw and processed materials, with a local scope, e.g. in an immediate proximity to such utterance.

1.2 Linguistic context and prosody

The definitions in the previous section highlight the crucial role that context has when interpreting meaning, but for TTS we care about predicting speech, and such, now we focus on how context is also crucial in influencing the spoken form. In this section we review some evidence from linguistics regarding the interaction between speech and the linguistic context. This is to show that when considering speech (in contrast to text alone), the linguistic context not only affects the interpretation of semantic or pragmatic meaning (as we saw in the previous section), but also the form that the speech will take for a particular utterance when in context. Most authors focus specifically on the interaction between context and prosody. Prosody is a suprasegmental dimension of speech, i.e. over or encompassing many segments, which are individual speech sounds or phones (Crystal, 2011). From a phonetics perspective, the most important acoustic prosodic features are fundamental frequency contours, speech rate and intensity. From a phonological perspective, it has been studied how these acoustic prosodic features can play a linguistic function as cues in the production and perception of speech. In this section we focus on studies that research the complex interaction between prosody and the **linguistic** context (although in Section 1.2.3 we inevitably consider how speakers play a fundamental role too when considering dialogue). We detect three points in which the linguistic context interacts with prosody, following the outstanding review on prosody and context by Cole (2015).

1.2.1 Structure

Prosodic cues can perform the function of marking linguistic structure. The linguistic context, being formed of utterances, is part of the linguistic structure. Therefore, elements of the linguistic context can be redundantly encoded by prosodic cues. Cole (2015) provides an extensive review on how the particular form that the prosody of an utterance takes depends both on the within-utterance elements (phrases, words, syllables) and the interaction among the utterances in the linguistic context. This is because it has been seen that multiple prosodic cues correlate with syntactic phrase boundaries and discourse segment boundaries (Cole, 2015, pp. 5-9), both of which pertain to the linguistic context structure.

The identification of specific prosodic cues in relation with the linguistic context has been shown further by Tseng and Su (2008) and Farrús et al. (2016), in terms of the position of a given utterance in a longer discourse unit. Tseng and Su (2008) analyzed how, for Mandarin read speech, the prosodic form of an utterance correlates with the position of that utterance in a spoken-paragraph (i.e. a group of utterances that when transcribed are given the orthographic boundaries of a written paragraph). Tseng and Su (2008) showed that fundamental frequency modulations are greater at the beginning of paragraphs, decreasing in the middle, and being smallest in final position. In terms of speech rate, a fast-to-slow pattern is seen from beginning to end of the paragraph. Farrús et al. (2016) analyzed English TED talks, finding significant prosodic differences among utterances depending on their position in a spoken-paragraph. They showed that mean fundamental frequency decreases for medial utterances in comparison to initial or final ones. Speech rate follows a similar pattern, being faster in the middle of paragraphs, and slower at the

beginning or the end.

Interestingly, the speech rate pattern found in Farrús et al. (2016) is not consistent with the one found by Tseng and Su (2008). This could be due to language differences or, more likely, style differences, as Farrús et al. (2016) analyzed semi-spontaneous speech, while in Tseng and Su (2008) analyzed read speech. These differences are also reported by Cole (2015, p. 6): for example, finding correlations between prosodic cues and syntactic boundaries for spontaneous speech is harder than for read speech. Even further, Cole (2015, p. 7) highlights speaker variability because speakers can encode syntactic boundaries through different prosodic cues.

1.2.2 Disambiguation

A crucial role of prosody is aiding the disambiguation of the meaning of an utterance in context (Cole, 2015, pp. 9-12). As explained in Section 1.2, context also plays this role, and therefore, it is natural that it interacts with prosody at this point. For example, prosodic prominence can help with the identification of the information status of a particular word or phrase in an utterance (Cole, 2015, p. 9). Information structure is inherently related to the linguistic context, as the determination of what is new or old will depend on previous utterances. In that respect too, the interpretation of an element as prominent is inherently relative to neighbouring units and therefore highly affected by the linguistic context (Cole, 2015, p. 9). Contrast is a particular case of emphasis that depends on context too. Different context utterances can bias different interpretations of contrast: for example, Stavropoulou and Baltazani (2021) used different context utterances to probe two possible types of contrast. Their analysis showed that each type of contrast presents different prosodic cues for Greek speech.

Two other clear cases in which the interaction between prosody and linguistic context affects meaning interpretation are irony and discourse markers, both for English. Bryant (2010) found that there are no inherent prosodic cues for ironic utterances. Rather, contrastive prosody between a context and a target ironic utterances was present significantly more than for other paired utterances. This results in, for example, the ironic target utterance being consistently longer in duration than the contextual one. Gravano et al. (2007) researched how linguistic context and prosody interact to disambiguate the function of the discourse marker *okay*. Multiple renditions of *okay* were extracted and labelled from video game dialogues by a first group of participants. The three main categories found were later presented to a second group of participants who listened them both in isolation and in context. The context included both the turn that contains the target words and the previous turn. Results showed that listeners' labels agreed very little when heard in isolation, but highly when heard in context. When considering the labels given in isolation, the authors found a correlation between prosodic cues present in the word and labels. However, when considering the labels in context, acoustic attributes of the context take precedence in the correlation, such as latency between turns or turn duration. The only exception to this interaction is that the final pitch tone of the target word correlates with the labels in both conditions.

Fox Tree and Meijer (2000) researched how linguistic context and prosody compete as cues to interpret syntactically ambiguous sentences. In a first experiment, a group of non-expert participants was recorded reciting memorized sets of three

sentences, where the one in the middle was the ambiguous one. Each ambiguous sentence was recorded twice, with two possible disambiguating contexts. Then, a different group of participants was asked to listen to the ambiguous sentences in isolation and instructed to predict the original context. Listeners were not able to accurately determine the original context. They conclude that, when producing the speech for ambiguous sentences, if context is available, potentially disambiguating prosodic cues are weak. In a follow up experiment, they first select a subset of the ambiguous utterances, the ones with the best results in the previous experiment, e.g. utterances for with the context was easier to predict from prosodic cues alone. These were presented in isolation to a different group of participants, and they were asked to paraphrase them, without knowing about the possible interpretations of context. Analyzing this new speech, they saw that prosodic cues were stronger, and usually with a tendency to one interpretation of context. This means that when there is no context, prosodic cues are stronger and can be used to derive a default context. In a final experiment, participants listened to the first set of ambiguous sentences again, but this time embedding them in each of the two possible disambiguating contexts. They were required to answer a question that implied one interpretation or another. The results showed that, in the presence of context, potentially disambiguating prosodic cues are ignored.

As well as when considering the interaction of linguistic context and prosody with respect to linguistic structure, style and speaker variability has also been found when considering their interaction for disambiguation. Mismatches between prosodic cues and information status has been found for spontaneous speech (Cole, 2015, p. 12). Similarly, speakers are not consistent in the use of pitch as a cue when producing ironic utterances (Bryant, 2010).

1.2.3 Dialogue

Dialogue is a special case where prosody and linguistic context interact tightly to encode structure and meaning as a function of the situational context, particularly the speakers' (Cole, 2015, p. 14). The turn is the primary unit of organizational structure in dialogue (Crystal, 2011). In dialogue, a sequence of turns, each of which can correspond to one or more utterances, can be related to the linguistic context. Turn taking is marked by prosodic cues. For example, Bögels and Torreira (2021) researched how speakers use both prosody and linguistic context to predict the end of a turn, by using a button press task for spontaneous telephone speech. Participants had to listen to the target turn in isolation or in context (together with the previous turn). Bögels and Torreira (2021) saw that context improved accuracy of turn-end prediction only for short turns, the intonational boundary being more important in the detection of a turn-end.

When considering interactive dialogue, the interaction between prosody and linguistic context is even more complex than for monologues. The compositional theory of intonation by Pierrehumbert and Hirschberg (1990) proposed that a particular speaker produces an utterance with a specific intonational contour to communicate the relationship between the meaning of the current utterance and the linguistic context, as well as the meaning of the current utterance and the speakers' mutual beliefs. One of the best examples of how prosody, linguistic context and speakers interact is speaker entrainment. As explained by Levitan and Hirschberg (2011), this

phenomenon has been observed in dialogue when a speaker progressively modifies his or her own speech, making it more (or indeed less) similar to the other speaker's speech. This process seems to be regulated by social dynamics such as the speaker's attitudes towards each other or the topic (Lee et al., 2010). Entrainment happens at turn changes, affecting fundamental frequency, speaking rate and intensity, the last being the most significant (Levitan & Hirschberg, 2011).

Prosody and linguistic context once again interact and compete, this time in the identification of the linguistic structure of dialogue. As well as pointed out in the two previous subsections, the strength of prosodic cues in dialogue to mark discourse structure is speaker dependent (Cole, 2015, p. 19). In terms of style, prosodic patterns present in spontaneous dialogue might be different from read speech, especially if speakers are not professional (Cole, 2015, p. 20). Finally, although entrainment unfolds as a consequence of the situational context, i.e the dynamics between the speakers, its consequences are encoded in the linguistic context, and are in fact possible to measure at this level, by comparing speech characteristics of consecutive utterances.

1.2.4 Why is this relevant for Text-to-Speech?

Our review of the research in the previous subsections aims to show some examples of the complex interaction between speech and linguistic context. Our review focused on the interaction with prosody only, as it is the most researched aspect in the literature. However, this does not mean that context does not affect segmental properties of speech, because, as stated by Cole (2015) all the levels of the hierarchical prosodic structure are affected by context, from segments to discourse units.

We believe that all this evidence is crucial to motivate the approach we propose: conditioning a TTS system on context to improve the synthetic speech. If speech encodes the linguistic context and interacts with it as a function of other elements such as meaning and the participants in interaction, then it seems reasonable to hypothesize that we can improve speech prediction by leveraging the linguistic context. All the linguistic evidence presented in this section points to the importance of linguistic context for speech which needs to be considered not only for training and synthesizing but also when evaluating TTS. Contrary to it, TTS systems have traditionally been trained and evaluated using isolated utterances, stripping any above-the-utterance context available.

It is important to notice how, for every point of interaction we found between speech and linguistic context, the authors highlight the presence of style and speaker variability. This will be considered in this thesis by developing a method to test our proposed approach and applying it to different speakers, for both read and spontaneous speech (within what is possible given our resources).

When starting this thesis research, we would have preferred to use dialogue data in order to research the distinctive phenomena described in Section 1.2.3. However, at the time of writing, there is no standard, large, high quality dialogue data set for TTS. Even though in Chapter 6 we collect real-world spontaneous dialogues, we show how difficult it is still, with state-of-the-art TTS technology, to handle such data. Therefore, most of our thesis works with monologue speech. Nevertheless, the methods we will propose can be taken as a starting point, which could feasibly be

extended into dialogue speech, once a standard data set becomes available.

Finally, although the research reviewed in this section focused on linguistic context, we know that the other dimensions of context reviewed in Section 1.1 also interact with speech. However, as explained in that section, linguistic context has characteristics that makes it apt for computational modelling and we limit the scope of this thesis to this type of context only. Considering the evidence presented in this section, even if we can only leverage a modest amount of context, we believe we can provide information that will improve synthetic speech.

1.3 Research overview and contributions

In this thesis we seek to answer three main research questions:

- **RQ1**: can we improve Text-to-Speech synthetic speech naturalness by conditioning on local linguistic context?
- **RQ2**: would the amount of any such improvement be greater for some speakers/styles than for others?
- **RQ3**: would we find significantly more improvements when evaluating synthetic speech in-context than for isolated utterances?

When we started the work that we will present in this thesis, to the best of our knowledge very little similar work had been done. The notable exceptions were Tyagi et al. (2020), which proposed to inform the inference of a variational autoencoder-based TTS system with previous synthesized utterances' features and Clark et al. (2019) who proposed an in-context listening test design. Other work was done in parallel to ours, which we review thoroughly in Section 9.2. Interestingly, most authors propose to leverage contextual information from past or future **text** to improve TTS. In contrast, a novelty of our work is that we researched the use of **both** acoustic and textual information, with a special interest on acoustics. Because of this decision, we limit the scope of our definition of local linguistic context further, to only the **previous** utterance with respect to a **current** utterance, as only these will be present as speech at synthesis time.

Figure 1.2 illustrates our definition of context in this thesis, where the previous utterance is the **context** utterance and the current one is the **target** utterance, for an example sequence of four utterances. Naturally, our research uses data sets that are not comprised of isolated utterances. Context and target utterance pairs are obtained from the sequence of utterances in the training and test data, such as a paragraph, a monologue or a dialogue.

This thesis has five experimental chapters where we report the experiments that answer these research questions. These chapters are organized in three parts. Part I comprises Chapters 4 and 5, where we develop a method to condition a TTS system on local linguistic context. Part II corresponds to Chapter 6, where we looked for alternative methods to incorporate context into TTS. Both Parts I and II answer **RQ1**. Part III is formed by Chapters 7 and 8. In Chapter 7 we take the best developed method and apply it to multiple data sets with diverse speakers and speaking styles, answering **RQ2**. Finally, in Chapter 8 we consider how to evaluate speech in context, answering **RQ3**.

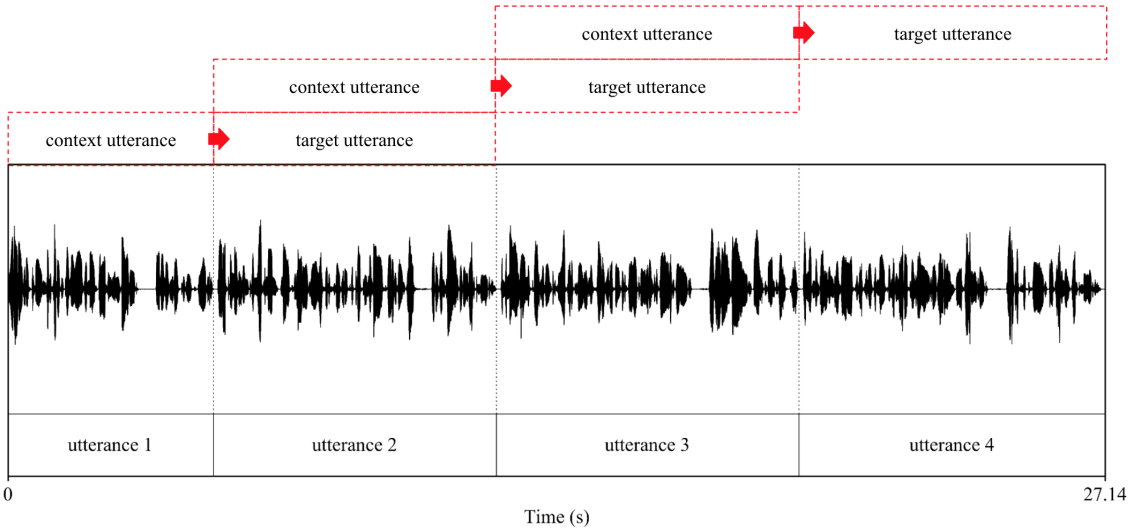

Figure 1.2: Illustration of how our definition of context is applied to data in this thesis.

Although every experimental chapter has its own research questions, hypotheses and contributions, the main contributions of this thesis, with respect to the research questions stated above, are:

1. A method to condition a TTS system on textual and acoustic representations, with a focus on the last one, extracted from local linguistic context.

2. A method to analyse natural speech re-purposed to evaluate synthetic speech in-context.

3. A demonstration that the improvements that our method can bring are speaker dependent, but only when using ground-truth context.

I have co-authored and collaborated on a few papers during the time of this PhD. Some of these papers are more relevant to this thesis than others. We list them here, indicating the extent of my collaboration and its relationship to the work reported in this thesis. There are three papers that are part of Chapters 4, 5 and 8 respectively, which will be referred into detail in those chapters:

- ***Oplustil-Gallegos**, P., and King, S. (2020). Using previous acoustic context to improve text-to-speech synthesis.* This paper was rejected for SLT 2020 for not including a comparison to textual representations from context, but, nevertheless, we considered that the reviews were good enough to upload it to ArXiv.

- ***Oplustil-Gallegos**, P., O'Mahony, J. and King, S. (2021) Comparing acoustic and textual representations of previous linguistic context for improving Text-to-Speech.*, published in SSW 2021.

- *O'Mahony, J., **Oplustil-Gallegos**, P., Lai, C. and King, S. (2021). Factors Affecting the Evaluation of Synthetic Speech in Context.*, published in SSW 2021. In this work, we expanded upon Clark et al. (2019) by experimenting

further with the design of in-context listening tests. I contributed in identifying which points from Clark et al. were important to expand upon, prioritizing aspects relevant for this thesis, and I extensively worked on the experimental design which we implemented in the paper.

This next set of papers are not directly related to this thesis, but were relevant in making some methodological decisions:

- *Fong, J.,* ***Oplustil-Gallegos****, P., Hodari, Z. and King, S. (2019). Investigating the Robustness of Sequence-to-Sequence Text-to-Speech Models to Imperfectly-Transcribed Training Data.* In this paper we tested how transcription errors in the training data can affect attention-based TTS models. This was the first paper I collaborated on starting my PhD, and it helped me to learn the use of a state-of-the-art architecture and critically analyze some of the disadvantages of attention-based architectures, which we subsequently abandoned after using them in Chapter 4. This paper was published in Interspeech 2019.

- *Dubiel, M.,* ***Oplustil-Gallegos****, P., Halvey, M. and King, S. (2020). Persuasive synthetic speech: voice perception and user behaviour.* In this paper we developed an interactive evaluation for TTS systems trained with spontaneous data. We saw the challenges of this kind of evaluation and this type of data, and given the results we decided to not attempt interactive evaluations in this thesis and to not use the same data set. This paper was published in CUI 2020.

- In the papers *Williams, J., Rownicka, J.,* ***Oplustil-Gallegos****, P. and King, S. (2020) Comparison of Speech Representations for Automatic Quality Estimation in Multi-Speaker Text-to-Speech Synthesis.* and
  Oplustil-Gallegos*, P., Williams, J., Rownicka, J. and King, S. (2020) An Unsupervised Method to Select a Speaker Subset from Large Multi-Speaker Speech Synthesis Data Sets* we experimented with different acoustic representations and their power to encode relevant aspects of speech. Given the positive results in this paper we make extensive use of Deep Spectrum features throughout this thesis. The first paper was published in Speaker Odyssey 2020 and the second one in Interspeech 2020.

This thesis is organized as follows: in Chapter 2 we review the Background knowledge relevant for this thesis, with a especial emphasis on how within-utterance context, in very general and different ways, has been considered throughout the history of TTS to bring improvements. In Chapter 3 we describe the Methodology used in this thesis in order to answer the main research questions stated above. As explained earlier, we then present five experimental chapters organized in three parts: Chapters 4 and 5 in Part I where we develop a method to conditions TTS systems on local linguistic context; Part II, Chapter 6, where we research alternatives methods; and Chapters 7 and 8 in Part III where we apply the best of our methods to diverse data sets and then evaluate in-context. Finally, in Chapter 9 we conclude by recapitulating the main results and contributions of the thesis, and by reviewing concurrent work similar to ours. This is followed by a discussion on limitations and guidelines for future work.

Chapter 2

Background

In this Background chapter we will describe the foundational knowledge on which this thesis is based. As a general rule, all the content in this chapter will be referred to, explicitly or implicitly, in more than one of the experimental chapters in this thesis. We refer to the most common followed practices in the TTS field. Recent research relevant to our work is referred to within the experimental chapters, with an additional field review of approaches similar to ours in the Conclusions chapter.

Briefly, we define some of the concepts to be covered in this Background chapter. Because, as we will see in the next sections, there is a contrast between concatenative frameworks and machine learning-based frameworks for TTS: when referring to the first one, we *build* TTS models, while when referring to the latter, we *train* models. We distinguish the use of the concepts *model*, *system*, *architecture* or *method*. A TTS *architecture* has not yet been trained or built, while a *model* has been. A TTS *system* includes the complete text-to-speech pipeline: a front end that processes the input text into a format that the TTS model can consume; the TTS model that generates speech or acoustic features; and, for the latter case, a waveform that is generated through a vocoder. Finally, a *method* entails a component or the ensemble of different components within a bigger architecture.

We start in Section 2.1 by describing the most common procedures to prepare data for TTS, including speech pre-processing and text analysis through the front end. In Section 2.2 we describe older frameworks for TTS: concatenative and Hidden Markov Model-based systems. Because all experiments in this thesis are based on deep learning-based TTS frameworks, in Section 2.3 we explain the fundamentals of neural network training. In Section 2.4 we start by describing the first deep learning-based architectures for TTS, to then make a detailed explanation of the state-of-the-art sequence-to-sequence architectures, which are the ones we use for the experimental work in this thesis. In Section 2.5 we discuss evaluation for TTS and we describe the listening test designs used in this thesis. In Section 2.6, we describe the approaches used in this thesis to obtain learnt representations from audio and text to encode context to then condition TTS models.

As stated in the Introduction chapter, we propose to improve TTS systems by explicitly conditioning them on local linguistic context. Throughout the Background sections we will emphasize how context has been progressively incorporated into TTS with every new framework. However, such context has always been **within** the utterance, while we propose to incorporate information contained in the **previous** utterance. We will give these remarks on the last Section of this chapter.

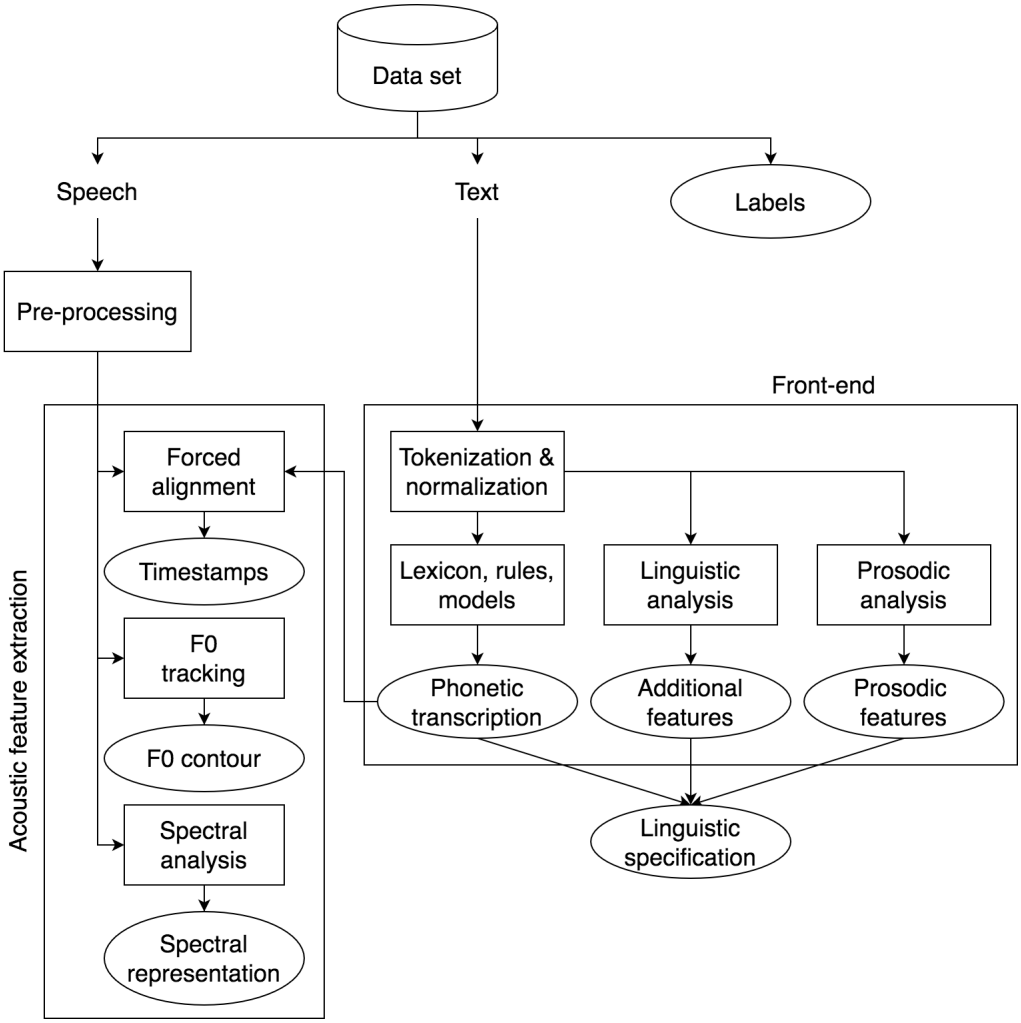

Figure 2.1: Summary of the main steps to process a data set into the necessary pieces of information to build or train a TTS system, described in Section 2.1. The squares symbolize processes while the ovals correspond to information that can be used directly by the TTS system.

2.1 Data processing

In this Section we revise the first steps that need to be taken in order to build or train any TTS system. In Section 2.1.1 we start by discussing how data sets are obtained and which are their most important characteristics. Fundamentally, a data set of matched speech utterances and text is required. Next, in Section 2.1.2 we explain how speech is pre-processed and acoustic features are extracted. Then, in Section 2.1.3 we describe the components of a basic front end, which is used to analyse the text. Figure 2.1 summarises the three stages of data preparation and how they are connected.

2.1.1 Data sets

Data sets are fundamental for any TTS system, regardless of the framework used to build or train it. Minimally, building a TTS requires matching pairs of speech audio and text utterances. The quality of the data will usually place a ceiling on

how good the synthetic speech quality can be (Taylor, 2009, p. 529). Low quality data sets include those with mismatches between the speech and the text, or where the audio has background noise, among other possibilities.

Data sets for TTS can be acquired, roughly, in two ways: recording data especially designed for TTS, or making use of existing data. Recording your own data set is costly, considering the time it takes to select and prepare text, record it, and the cost of remunerating the speaker(s). In comparison, using freely available data is less costly but it can be very variable in terms of overall quality, presenting other challenges to the researcher.

Every data set can be usually described through some basic but very relevant characteristics: is it comprised by one speaker only (*single speaker*) or multiple ones (*multi-speaker*); what is the native language of the speaker and the language of the speech (which might not necessarily match, for example, for L2 speakers); what is the speaking style (for example, if it is emotional or neutral, or read speech vs spontaneous speech); what is the content and extent of the linguistic material, also known as *domain*; and which were the recording conditions (for example, studio recorded or not). The main advantage of recording a data set that has been especially designed for TTS is that it is possible to tailor some or all of these characteristics, either for research purposes or for a designed application. In contrast, when making use of existing data sets, all these characteristics are already fixed.

In this thesis we only use data sets already available, free, and appropriately licensed for research (with the exception of Chapter 7, where we present work carried on during an industry internship, where the company bought the data sets). We selected the data sets most appropriate for our research questions among the available ones. We only work with English data, as it is not only the language for which the most available data exist, but also other resources that are language dependent (such as large pre-trained language models as the ones described in Section 2.6.1). We will describe and summarise all the data sets used in this thesis in Section 3.2.

2.1.2 Speech pre-processing and acoustic feature extraction

Two consecutive stages are usually applied to the data set speech: first, it is pre-processed in order to convert the waveform files into a form that is suitable for the second stage, acoustic feature extraction. These features will be the ones used by the TTS system rather than the original waveform (with some notable exceptions as unit selection, explained in Section 2.2.1, and systems like WaveNet, which we briefly comment on in Section 2.4).

Which pre-processing steps are taken will depend on what the data set requires, but some of the most common steps are: utterance segmentation, power normalization, silence removal and end-pointing, resampling and noise filtering. These actions will effectively modify the waveform material which then will be used as a source for the acoustic feature extraction. End-pointing and silence removal is needed because some speech-based systems struggle with the silent regions of the speech (such as some forced alignment systems, see below). Noise filtering might be required if the data set has background or other types of noise. If the data set is formatted as speech-text pairs at the paragraph level, chapter or other kind of long unit, it needs to be segmented into utterances. Utterance segmentation is fundamental because all TTS frameworks model text-speech pairs at the utterance-level.

Acoustic feature extraction will depend on the TTS framework and research purpose. Here we revise only the most common acoustic features that are extracted from the pre-processed speech waveforms: timestamps, fundamental frequency contours, and spectral representation.

Forced alignment is used to obtain timestamps for the speech with respect to a phonetic transcription (Taylor, 2009, p. 479). An acoustic model of the language (usually pre-trained) together with a restricted language model (as the linguistic content of the utterance is already known) are used to find time aligned boundaries for every phonetic symbol in a given utterance. Timestamps can be used later, for example, to train duration models that predict the duration of an input symbol at synthesis time. In this thesis, unless stated otherwise, we use the Montreal Forced Aligner (McAuliffe, Socolof, Mihuc, Wagner, & Sonderegger, 2017).

The fundamental frequency (f0) is an acoustic property of a sound, which corresponds to the number of waveform cycles per second (Taylor, 2009, p. 149). An f0 contour is the result of tracking the fundamental frequency of sound with respect to time. In this thesis, unless stated otherwise, we use Praat (Boersma, 2001) via a Python library (Jadoul, Thompson, & de Boer, 2018) for f0 extraction. A detail description of the fundamental frequency tracking algorithm can be found in (Boersma et al., 1993). F0 contours can be used explicitly to train some TTS models to predict f0 jointly with spectral representations.

While the forced alignment and f0 extraction can be considered optional, every TTS framework relies on a representation of the spectral information of the speech. Different spectral representations can be extracted (Taylor, 2009, p. 350). The most common spectral representations used in TTS have changed as the predominant framework, and therefore, we will review them in detail together with the frameworks in the subsequent sections of this chapter.

2.1.3 Linguistic feature extraction and labels

A front end is a pipeline of modules that perform a linguistic analysis of the data set text, resulting in a linguistic specification. This is a resource needed both during building/training systems and synthesis/inference time. For this reason, front ends tend to be more TTS specific than the toolkits used for pre-processing speech or extracting acoustic features.

Minimally, a front end should output a phonetic transcription for an input text, to map the orthographic text to a string of phonetic symbols that represents the speech sounds or segments (sometimes including suprasegmental information, such as stress). Although some state-of-the-art TTS architectures claim comparable results modelling graphemes instead of phonetic transcriptions, e.g. (Łańcucki, 2021), the standard is still to use phonetic transcriptions.

Phonetic transcriptions of words can be obtained through three methods: (1) a lexicon, (2) rules, or (3) machine learning-based models. A lexicon stores a list of words with their phonetic transcription(s). Rules are used to compactly express the grapheme-to-phoneme relationships of a language, especially for those with a mostly unambiguous correspondence between letters and sounds (such as Spanish). Rules are usually hand-written, while lexicons tend to be a combination of rule-generated output with manual corrections, and most commercial systems use both (Taylor, 2009, p. 209). Their main disadvantage is their generalization power: if a word is

not in the lexicon or if it does not fit the available rules, then no accurate phonetic transcription can be obtained. Machine learning-based models are trained on data for the language (which, ideally, has not been automatically generated, such as a lexicon), and therefore, should have the greatest generalization power among these three methods. With the exception of phonetic feature-based methods, e.g. (Wells & Richmond, 2021), these tend to be language-dependent.

An important clarification to bear in mind is that the terminology used in the TTS field when referring to the units of the phonetic transcription is not as strict as in the linguistics field (Moore & Skidmore, 2019): authors interchangeably use "phone", "phonetic" and "phoneme", which is not correct from the phonological and phonetic perspective. Because TTS "phonetic" transcriptions are, as we saw above, obtained from text (generally) without inspection of the corresponding speech, its units are not "phones", as a phone symbol represents a sound. Actually, the units of the TTS "phonetic" transcription are closer to the concept of "phoneme", as transcriptions are usually extracted considering text alone, and include only one pronunciation that is considered "standard" of a word, with emphasis on phonemic contrasts rather than allophonic variation. In this case, there is an obvious contradiction in calling it "phonetic" transcription. Despite that, in this thesis we follow the field's convention and we refer to it as a phonetic transcription.

Most methods to obtain phonetic transcriptions do so at the word-level. Therefore, text is pre-processed by finding the words in it through tokenization. For English, most of the time words can be identified by simply splitting the text at white space. Additionally, text normalization should be performed if needed, where non-alphabetic symbols (such as numbers) or non-standard text (such as abbreviations) are expanded into their full textual form. Some systems might include additional steps such as Part-of-Speech tagging to aid disambiguation of homographs.

When processing the text with the front end, it is common to augment the resulting phonetic transcription with additional, simple text-derived symbols such as: word boundaries, punctuation marks, and start and end of sentence symbols. Some languages might benefit from other additions, such as syllable boundaries. Moreover, some authors propose additional information extracted from the text through linguistic analysis, such as syntactic and morphological analyses (e.g. Song et al. (2021)). It is also common to include prosodic features that can be derived from text, such as phrase boundaries and breaks or pitch accents, e.g. (Suni, Kakouros, Vainio, & Šimko, 2020), particularly important for tonal languages. Although this additional linguistic information can benefit TTS systems in general, the improvement is often marginal and automatic analyses of such information are not always accurate (Yasuda, Wang, & Yamagishi, 2021).

Besides features that can be derived from the input text through the front end, there are other additional inputs such as labels. Labels are usually at the utterance level and describe some overall characteristic of the speech-text pair. The most frequent use of labels is to provide speaker identity and speaking style or emotion. Labels are used both when building/training a system and when synthesizing/performing inference with it. They can be obtained from metadata or by additional analysis of the audio or text, or through manual labelling.

Additional linguistic analysis was more commonly used with pre-deep learning TTS frameworks that relied heavily on text-derived features, while state-of-the-art TTS architectures tend to make use of *learnt* additional inputs (see Section 2.6),

although labels are still commonly used. We observe that there has been an inverse relationship between the complexity of the front end and the evolution of TTS architectures: with the shift to deep learning-based frameworks, front ends have been simplified, as it is expected that the TTS architecture will model information that before would have been explicitly provided to the model in the form of input features (Yasuda et al., 2021).

In this thesis we make use of *learnt* representations obtained from both text and speech in order to improve TTS systems. As such, we do not focus on the front end: we use a basic linguistic specification that comprises phonetic transcriptions and the simple text-derived features described earlier (see Section 3.2).

2.2 Older Text-to-Speech frameworks

In this section we will quickly review the TTS frameworks that prevailed before the shift to deep learning: concatenative, and Hidden Markov model (HMM)-based TTS. These frameworks have now been significantly outperformed by deep learning, both in *robustness* (i.e. frequency of mistakes in the synthesized speech) and *generalization* (i.e. high quality speech for out of domain text) capabilities (with respect to concatenative frameworks), and naturalness (with respect to HMM-based ones). Although in this thesis we use only deep learning-based frameworks, reviewing pre-deep learning ones will allow us to demonstrate how there has been a trend of improving TTS by incorporating more and more within-utterance contextual information.

2.2.1 Concatenative Text-to-Speech

Concatenative frameworks for speech synthesis are based on slicing speech units from a database of pre-recorded, annotated, speech, in order to compose a novel sequence of speech units for the given input text (Taylor, 2009, p. 424). To build a concatenative system, first, text materials are selected to be recorded to obtain matching speech samples, collecting both into a database. Then, the recorded speech and text are processed as described in Sections 2.1.2 and 2.1.3, saving all extracted information into the same database. At synthesis time, the input text is analysed and a search is performed to retrieve from the database the speech units that best correspond to the linguistic specification of the input text, to produce a new utterance. Concatenative systems are based on the concept of the unit. Although other units had been proposed (Kishore & Black, 2003), the diphone was the most common. A diphone is a unit that comprises the second half of a phone along with the first half of the following one (Taylor, 2009, p. 412). The edges of the diphone are placed at the most stable portion of a phone - the middle - to facilitate concatenation. Moreover, because the diphone's middle point is the transition between two phones, it naturally captures coarticulation.

Unit selection systems were the second-generation of concatenative systems. For an overview of the building process and the synthesis process with these systems, see Figure 2.2. Text selection from external text sources was performed to obtain a recording script to elicit a relatively large amount of speech (more than for the first-generation, diphone systems), aiming to obtain multiple, diverse examples per diphone type. The assumption was that having many speech units as potential

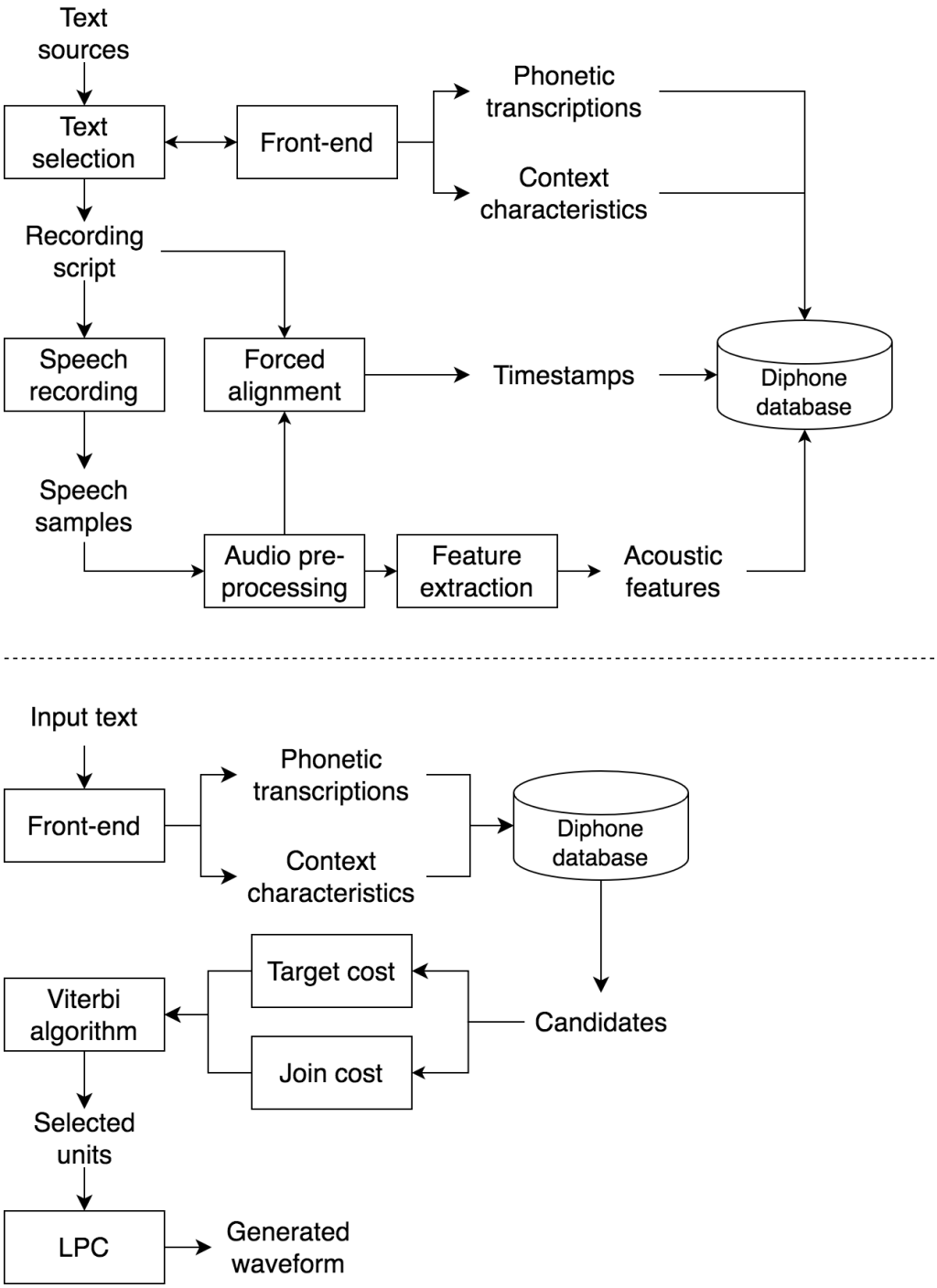

Figure 2.2: Summary of the components to build a Unit Selection system (top half) and to synthesize with it (bottom half).

candidates to re-arrange into a new utterance would increase the chances of finding a sequence of units that concatenates more smoothly (avoiding signal processing adjustments), and units that would be a better match for the input text (Taylor, 2009, p. 486). The matching text is pre-processed and analysed by a front end as it was described in Section 2.1.3, and the speech processed as described in Section 2.1.2. Phonetic transcriptions are obtained for all the recorded texts, and forced alignment is performed to obtain timestamps for every unit.

In theory, timestamps obtained via forced alignment and phonetic transcriptions from the database are the minimum information needed to perform synthesis with a unit selection system. In practice however, in order to select units that match as closely as possible the linguistic specification and that smoothly join with each other, **context**-dependent diphones were proposed (Strom, Clark, & King, 2006). This means that not only the diphone identity, but also a set of characteristics regarding the within-the-utterance context of the diphone is taken into account. At synthesis time, a searching algorithm will consider this information to find the best possible units.

In Festival, for example, when using the multisyn back-end (Clark, Richmond, & King, 2007), a search algorithm based on a target cost and a join cost is applied. The target cost is a model of the within-the-utterance context: it is designed to match as closely as possible the units and contexts analysed from the input text with those of the candidate units. It comprises the following criteria: the position of the diphone in the syllable, the word and the phrase, the left- and right-adjacent phonetic contexts, the stress assigned to the syllable and the part-of-speech of the word. The final target cost is a weighted sum of these criteria. The join cost is calculated in order to acoustically compare the candidates when put together, scoring better those that naturally transition more smoothly when concatenated. It measures discrepancies in the fundamental frequency, spectral frequency and log energy. The target and join cost are combined and an overall cost is assigned to every sequence. The Viterbi algorithm is used to select a final sequence that is globally optimal for the input text.

Unit selection systems were better than diphone systems, and they can still compete with modern TTS systems for limited domains, where having sufficient data can lead to very natural speech. In fact, the data itself is the "model" in unit selection. Therefore, any problem in the data or the lack of it, is the root cause of the disadvantages of these methods (Kuligowska, Kisielewicz, & Włodarz, 2018): they are not robust to low quality data, and they are hard to scale and to control in comparison with machine learning-based frameworks.

2.2.2 Hidden Markov Model-based Text-to-Speech

With Hidden Markov Model (HMM)-based TTS, rather than using the data as a "model", statistical models are learnt from the data (Taylor, 2009, p. 447). These models were not learnt from speech waveforms but rather a set of acoustic features: speech is decomposed into time-aligned spectral, fundamental frequency and aperiodic energy (including deltas and delta-deltas) frame vectors using a vocoder. During training, the models learn to map text input symbols to acoustic features. An HMM combined with a Gaussian Mixture Model (GMM) are trained for every phone in the language. For every HMM state there is an associated GMM. The

GMM is a combination of Gaussian distributions, each one with a mean and a variance for a set of acoustic features. Acoustic features are aligned to phones using the Viterbi algorithm or the Baum-Welch algorithm. At synthesis time, the models can generate acoustic parameters given a new input text. In order to create a speech waveform from these acoustic parameters, the vocoder is used. To train the models, we can use a data set obtained in the same way as described for a unit selection system in the previous section.

In practice, it is desired to have models for every phone in its context (similarly to the context-dependent diphones in unit selection). These **context**-dependent models are defined by taking into account multiple linguistic features such as position in sentence, stress, phone neighbours, etc. In fact, systems like HTS use about 25 features as their definition for context-dependent phones (Zen, Tokuda, & Black, 2009). Similarly to unit selection, it is very likely that only for a limited number of these contexts there are multiple training examples available, while for the rest there is only one or none (Zipfian distribution). We cannot estimate the distribution of the acoustic features for a phone with one or no samples. To solve this, models can be grouped to share model parameters with those whose contexts are linguistically similar to each other (Zen et al., 2009). To decide on the linguistic similarity of the within-the-utterance context, the context characteristics are clustered using, for example, decision trees. Decision trees are in practice a fundamental part of these systems, and are effectively the model of the within-the-utterance context (Zen et al., 2009). In this case, instead of a fixed model of context, as in the target cost for unit selection, the context definition is truly data dependent for each phone, as decision trees are trained with data. Linguistic characteristics from the context that are not relevant for the data are effectively ignored while training the decision trees, creating a model of context tailored for every phone.

At synthesis time, the input text is analysed by the front end obtaining the phonetic transcription and the linguistic context for every phone. This is input to the decision tree, which points to the parameters for the HMM-GMM models to use, and decoding is performed. In its simplest use, we can sample the mean of the GMM to generate the acoustic features. The generated features are run through the vocoder and a waveform is generated. During synthesis, the generation of every acoustic feature is subject to the HMM assumption that each observation depends only on the emitting state (regardless of past or future observations). Deltas (and delta-deltas), which are then leveraged by the vocoder, are also generated to alleviate this assumption. To ensure that deltas are statistically related to the acoustic features the maximum likelihood parameter generation (MLPG) algorithm was proposed (Tokuda, Yoshimura, Masuko, Kobayashi, & Kitamura, 2000), improving the smoothness and naturalness.

This framework brought some significant advantages over unit selection, including that systems could be trained with very limited amounts of data ($\sim$ 20 minutes) and were more robust than unit selection systems (Kuligowska et al., 2018). The main advantage however, is that because the model is operating in an acoustic feature space, multiple transformations can be performed that allow, for example, for speaker or style conversion or adaptation.

The introduction of vocoders into the TTS pipeline made the output generated with a characteristic "buzzy" quality due to the way in which these systems model signals. Another major disadvantage is that the speech, although very intelligible,

tended to have a very neutral style (as the models tend to produce average speech parameters), affecting the naturalness and expressivity of the synthetic speech.

2.2.3 Mel scale spectrum and classical vocoders

The statistical properties of the HMM-GMM models required the use of a certain type of spectral features. The most common choice was mel generalized cepstral coefficients (MGCs). These coefficients have the advantage of being relatively statistically independent from each other. When modelling them with the GMM covariance can be ignored, reducing the number of parameters to be trained.

The MGC extraction pipeline starts by applying pre-emphasis to adjust spectral tilt. The Short-Term Fourier Transform (STFT) is used to obtain a frequency representation of the audio. This is applied to small windows of the signal, defined by a window length of audio samples. That window slides through time by a given hop length, usually smaller than the window length, allowing a certain overlap, to avoid missing any information. The linear frequencies are warped through a frequency scale more similar to the human audition of frequencies, with a higher resolution in lower frequencies (Tokuda, Kobayashi, Masuko, & Imai, 1994). Phase is discarded and the log is taken. The spectrum is highly correlated, so the STFT is applied once again to obtain decorrelated coefficients, known as cepstrum.

While the HMM-GMM model is trained to generate MGCs, fundamental frequency and aperiodic acoustic features, to complete the speech synthesis process, it is necessary to reconstruct a waveform from these acoustic features. A vocoder can be used to both extract these features from waveforms and recreate waveforms from the same acoustic features. Vocoders can crucially affect the final quality of the reconstructed speech.

2.3 Deep learning fundamentals

Neural networks (NNs) are a set of models that are designed to learn a task. The learning stage of a neural network is called training. Once a neural network is trained, it can be used to perform the task it was trained for, for inputs unseen during training, i.e. inference. There are two dominant learning approaches to train neural networks: supervised and unsupervised. To train a neural network in a supervised fashion, data organized in input-output pairs is required, and most commonly, learning is formalized as either classification or regression. When classifying, the neural network is trained to label the input as one of a closed set of possible target categories, such that the neural network will output the most likely one. In contrast, for regression, the neural network is trained to map an input to an output that is a continuous vector. Unsupervised learning aims at finding hidden patterns in the data without the need of matched input-output pairs during training. In this section, we focus on supervised learning.

TTS performed using neural networks is usually formalized as supervised regression, mapping input text to continuous acoustic values (just as we described in Section 2.2.2 for HMM-GMM-based TTS). While in Section 2.4 we will focus on how neural networks are used for TTS, in this section, we will focus on a general description of neural networks used for regression: in Sections 2.3.1 and 2.3.3 we describe the basic components of the neural network architectures and in Sections

2.3.2 and 2.3.4 we describe how neural networks are trained. Because in the next section we focus on a general description of neural networks, we base our explanation on networks that take as training data rank-1 tensors. In Section 2.4.2 we will explain how neural network training and architectures are can be extended to sequences.

2.3.1 Basic architecture components

Neural networks are composed of layers of neurons and activation functions. An input x is usually a vector with a number of features d. A neuron, or unit, computes the weighted sum of its input(s) and a bias, where the resulting output y is transformed by an activation function. Weights and activation functions regulate the information flow through the network. During training, the values for the weights, also known as parameters, are automatically learnt (see next section). For complex task such as TTS, the mapping that must be learnt between the input and the target outputs is not linear. While the weights perform linear transformations, activation functions introduce non-linear transformations into the model, and are usually applied to every neuron of a particular layer. Activation functions also help to maintain the numerical values that flow through the neural network within a desired range that facilitates training.

A neural network is a stack of layers (most commonly) connected sequentially. A neural network has at least three layers: an input layer, that receives the input data; one or more hidden layers, stacked in between the input layer and the output layer, that receive and feed information to other layers; and an output layer that generates the output. A neural network with multiple hidden layers is known as a "deep" neural network. A layer is defined by an input dimension and an output dimension. The input layer of the network will usually have an input dimension equal to d, with an arbitrary output dimension to which the input data will be transformed. The output of the hidden layers is usually called as hidden representation, as it is expected that each hidden layer will find an underlying pattern in the training data.

2.3.2 Training neural networks

The set of all the trainable parameters of a neural network will be denoted by Θ. The total number of parameters will depend on the neural network's architecture: how many layers, their sizes, types, etc. Training consists of automatically and iteratively adjusting parameters of the neural network given a training data set of input-target examples $\langle x, y \rangle_i$. The goal is to reach a set of parameters for which a predicted output $\hat{y}_i$ is as close as possible to the target y_i when the input is x_i.

Before training starts, parameters have no assigned values yet. An initialization strategy must be applied, such as random initialization. First, $\hat{y}_i$ is calculated given x_i for the current parameters. This is known as a forward-pass. When $\hat{y}$ has been calculated for every sample in the training data, an epoch has been achieved. Next, a loss function L is applied to calculate the error of the output generated by the current parameters, comparing $\hat{y}_i$ to the targets y_i. In regression models, a common loss is the Mean Squared Error (MSE) shown in Equation 2.1, where N is the total number of examples.

$$L(\Theta) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{2.1}$$

The ultimate goal of the training process is to find the optimal values for Θ, that will reduce the loss to the minimum for the training data. Such trained network should later output good predictions for inputs unseen during training. To iteratively improve the parameters with respect to the loss and considering the error, after every epoch the parameters are updated given Equation 2.2. For every parameter we need to determine how to update it regarding the direction of change (increase or decrease) and by how much. The hyperparameter α in Equation 2.2 determines the latter. The direction of change is obtained by calculating the derivative of the loss function with respect to the current Θ.

$$\Theta \leftarrow \Theta - \alpha\frac{dL(\Theta)}{d\Theta} \tag{2.2}$$

The derivative of the loss with respect to Θ is composed of the partial derivatives of each parameter in the neural network. Partial derivatives are obtained by differentiating the terms of the parameters that are connected in the network through the chain rule, starting from the error obtained by the loss function with respect to the targets, back-propagating the calculation, to the input layer. Partial derivatives with respect to the loss are gathered in a gradient vector, and Equation 2.2 is applied to update the parameters.

Once the parameters have been updated, a new epoch starts and the process is repeated, slowly improving the parameters to reduce the error of the network. This training algorithm is known as Gradient Descent when we update the parameters taking into account all training data. In practice though updates are performed through different mechanisms. For example, Stochastic Gradient Descent performs the update individually for mini batches in the training data.

We need a criterion to decide when to stop training. One option is to define a maximum number of epochs or, training can be stopped when convergence has been reached: if the loss does not change anymore or if it has reached a value close to zero. For a formalized version of the training algorithm described in this section, see Algorithm 1.

2.3.3 Types of layer

In this section we summarise some of the types of layers that are referred to throughout this thesis. Because all deep learning implementations in thesis are based on the PyTorch deep learning library for Python (Paszke et al., 2019), the descriptions here are heavily based on the library implementations.

Embedding layer: an embedding layer is a trainable look-up dictionary defined by a vocabulary size and an embedding dimension. The dictionary returns an embedding for an index. Each embedding is a set of weights. The embeddings are initialized randomly. It is very common to use this type of layer as the first step in a network to embed symbolic data into a continuous space. For example, each phoneme in a language can be given a unique index, with a vocabulary size equal to the number of phonemes. Then, for any given index, the embedding layer will return the corresponding learnt embedding.

Algorithm 1 Simple application of Stochastic Gradient Descent

```
network = Architecture()
network = network.init(random)
lrate = 0.1
epochs = 100
x = array([inputs])
y = array([targets])
assert len(x) == len(y)
batches = dataloader(x, y)
for epoch in epochs do
  for xbatch, ybatch in batches do
    yhat = network.forward(xbatch)
    error = loss(yhat, ybatch)
    gradient = network.backward(error)
    network.update(lrate, gradient)
  end for
end for
```

Linear layer: a linear layer is the most basic type of layer. It is feed-forward (i.e. information only moves forward) and fully connected, and applies a linear transformation to its input. An input and output dimension must be defined, which will determine the number of weights of the layer.

Recurrent layer: a recurrent layer takes two inputs: an input vector and a hidden state. The latter corresponds to the output of the same layer in a previous timestep. Therefore, recurrent layers are suited to modelling sequences. For example, each phone in the phonetic transcription of an utterance can be transformed into indexes. After retrieving embeddings for it, each one can be passed, one by one, through a recurrent layer. If the second phone is the input to the recurrent layer at the current time *t*, then the hidden state of the same layer for phone *t-1* is used as input too. Recurrent layers are therefore not feed-forward. A recurrent layer makes a weighted sum of the input vector and the previous hidden state to generate a new hidden state for *t*, with a pre-defined dimension. Recurrent layers can also be bidirectional.

Although recurrent layers can help to model temporal dependencies, when calculating the partial derivatives, the elements that are fed-back need to be considered. The multiplication of so many terms can lead to what is known as "vanishing gradients", resulting in very small numbers, approaching to zero. Near-zero gradients result in very small updates of the parameters, especially for layers towards the input, stagnating learning. The extended auto-regressive calculation also results in increasing training times. Truncated back-propagation, Rectified Linear activation functions or gradient clipping can be applied to mitigate on these issues.

Long Short Term Memory and Gated Recurrent units: although not a type of layer, LSTMs and GRUs are special types of units that have been designed to improve on the issues described above for recurrent layers. Both of them incorporate gates that learn to regulate which steps of the auto-regression are relevant for parameter updating, decreasing the number of terms taken into account for back-propagation.

1D Convolutional layers: convolutional layers are most commonly used in the computer vision field, but have in recent years become popular for TTS, especially

in its 1D implementation. A convolutional layer applies a moving filter or kernel for which cross-correlation to the input is calculated. The weights within the kernel are learnt during training. When defining this layer therefore, a size for the kernel is determined, as well as the a stride (i.e. the step size when moving the kernel). Kernels can be dense or dilated, e.g. the filter is not applied as a dense block but rather with spaces in between.

Residual connections: although not a type of layer, residual connections are also used to keep the flow of information through the network and avoid vanishing gradients. They are a very simple mechanism that consists of keeping a certain vector through out the network computations. For example, a certain input that is passed through a layer is not only processed by the layer, but also directly summed to its output, as in Equation 2.3.

$$output = layer(x) + x \tag{2.3}$$

2.3.4 Further training concepts

In Section 2.3.2 we described the standard procedure to train neural networks through Gradient Descent. In this section we provide further detail on some of the most common issues during training and standard practices to handle them.

The loss function will have one or more global minimum, that is, the optimal set of parameters, and multiple local minima that should ideally be avoided. Depending on how parameters had been initialized and the chosen learning rate, Gradient Descent might get "stuck", e.g. the gradient will stop changing and training will converge without having reached the optimal parameters to minimize the loss function to a global minimum.

One way to overcome sub-optimal convergence is through warm-up, which consists of introducing a scheduled learning rate. The schedule has three phases: at the start of training the learning rate is at its maximum value; during the cool-down period, the learning rate slowly decreases until reaching a minimum value; and finally, the value is kept constant until training finishes. A larger learning rate at the start of training can help to avoid local minima by making larger updates, progressively making smaller ones as the schedule is followed.

Another common problem when training a neural network is that it might overfit to the training data. This means that the neural network's parameters memorize the data instead of learning generalizable patterns, performing poorly for unseen inputs during inference. One way to improve generalization is by introducing controlled perturbations in the neural network through dropout (A. Zhang, Lipton, Li, & Smola, 2021). Given a probability, dropout will randomly set to zero some of the outputs of a particular layer before passing them to the next one during the forward pass. By forcing the network to rely on different connections at different points of training, it becomes less likely that memorization occurs.

As referred to in Section 2.3.3 vanishing gradients can happen, and conversely, exploding gradients can also be a problem when training a deep neural network. Normalization methods can be used to keep parameters within a certain range and improve training. There are many normalization strategies, but two of the most common are batch and layer normalization. Batch normalization consists of calculating the mean and standard deviation of every element in a batch, independently

of the feature dimension. Layer normalization performs the same operation but over the feature dimension, independent of the batch samples.

Multiple algorithms have been proposed to improve Gradient Descent. One of the most popular at the moment is Adam, an adaptive method for stochastic optimization (Kingma & Ba, 2014). When optimizing with Adam, instead of having a unique learning rate, every parameter has a particular adaptive learning rate, depending on the local gradient. Using Adam makes training faster.

Finally, we would like to stress the large number of decisions that need to be taken when designing a neural network: how many layers and of which dimension, learning rate, optimization algorithm and its hyperparameters, dropout, normalization, initialization, among many others. Ideally, for a particular experiment, we would try architectures with different combinations for all these hyperparameters and design choices to then select the one that gives the best results. In practice this is infeasible due to time and computational cost constraints, and therefore we need to rely either on heuristics, or designs and choices reported by other researchers as successful.

2.4 Deep learning-based Text-to-Speech

Deep learning became the most popular approach through wider access to GPUs and memory. As mentioned before, because HMM-GMM-based frameworks are formalizing a regression task too, the first approaches to introduce neural networks into TTS were simply replacing the decision tree with a neural network. However, instead of having one HMM-GMM model per phone, only two neural network models are trained with all the available data: one to predict duration and one to predict acoustic features. The duration model first predicts how many frames correspond to each phoneme in an utterance, so that the acoustic model can predict each acoustic feature frame. This is why we will call this framework "frame-level" neural network-based TTS. To synthesize the final waveform, the classical vocoders described in Section 2.2.3 were used. We will describe this framework in Section 2.4.1.

A new approach to learning and generating acoustic features was proposed through more advanced neural network architectures that can take into account whole sequences, called sequence-to-sequence models. In contrast to "frame-level" neural networks, these architectures allow, through different methods, the prediction of an acoustic feature frame while taking into account more than one element in the input sequence. Moreover, these systems were combined with vocoders that, unlike before, are neural-networks ("neural vocoders"), which have become the state-of-the-art approach. All the experiments in this thesis follow this approach, and we review this framework in Section 2.4.2.

Because our thesis focuses on improving the prediction of intermediate acoustic features rather than on improving neural vocoders, we finish this section with a broad explanation on how those work and a slightly more detailed description of the particular neural vocoder we use for our experiments (Section 2.4.3). Since the release of WaveNet (Van den Oord et al., 2016), end-to-end systems, i.e. TTS architectures that do not rely on intermediate acoustic features but rather map text to a waveform directly, have also been an important topic of research in TTS. However, we will not refer further to these systems in this thesis, because we instead use the two-step framework that requires the inclusion of a neural vocoder.

2.4.1 Frame-level neural network-based Text-to-Speech

The earlier frameworks to do TTS through neural networks used relatively simple architectures and, as mentioned before, were heavily based on the HMM-GMM framework described in Section 2.2.2. The usual approach would require training two, independent, neural networks: a duration model, which learns to predict how many frames to produce for every phone in an utterance, and an acoustic model that learns to predict the frame-level acoustic features.

In this section we will describe how these models are trained and used to generate synthetic speech, using many of the concepts described for general neural network training on Section 2.3. We start by giving an overview of these two models in Section 2.4.1.1 and 2.4.1.2, which are summarized in Figure 2.3. In Section 2.4.1.3 we describe in more detail the main architectures used for these models.

It has been shown that using a neural network as a regression model can generate synthetic speech with significantly better naturalness than with HMM-GMM-based TTS (Zen, 2015), even if waveform generation still depends on a non-neural vocoder. The main difference between HMM-GMM-based and neural network-based frameworks is that while for HMM-GMM-based TTS one tied-model is trained for each subset of context-dependent phones, while when using neural networks, only two models are trained, and both can leverage all the available training examples. This advantage can be further exploited by multi-speaker, multi-language models (B. Li & Zen, 2016).

At that moment in time, a clear disadvantage of these models with respect to HMM-GMM ones, was the time it took to train and synthesize, but this has been greatly improved since. The main disadvantage of neural network-based TTS is that these models require large amounts of data to be trained (in comparison with HMM-GMM models), and with the evolution of these architectures instead of easing this requirement, even larger data sets are required.

2.4.1.1 Duration model

The input for the duration model is a linguistic specification for every phone, obtained through a front end (just as explained in Section 2.2). The linguistic specification is encoded first into a numerical format apt for neural network training. The questions used for the decision tree models described in Section 2.2.2 can be used for this purpose, turning binary questions into vectors through one-hot encoding.

The training target of the duration model is a vector with the number of frames for the phone, which is derived from the timestamps obtained through forced alignment. Each vector for every phone is learnt, one at the time. Training can be performed using Stochastic Gradient Descent, as explained in Section 2.3.2, to minimize the MSE loss. At synthesis time, the duration model will output the number of frames for the phone. Because at synthesis time this information is needed for the acoustic model to generate the acoustic features, the duration model needs to be run first.

2.4.1.2 Acoustic model

During training of the acoustic model, the linguistic specification for a phone is upsampled to as many frames as the forced alignment timestamps indicate. The

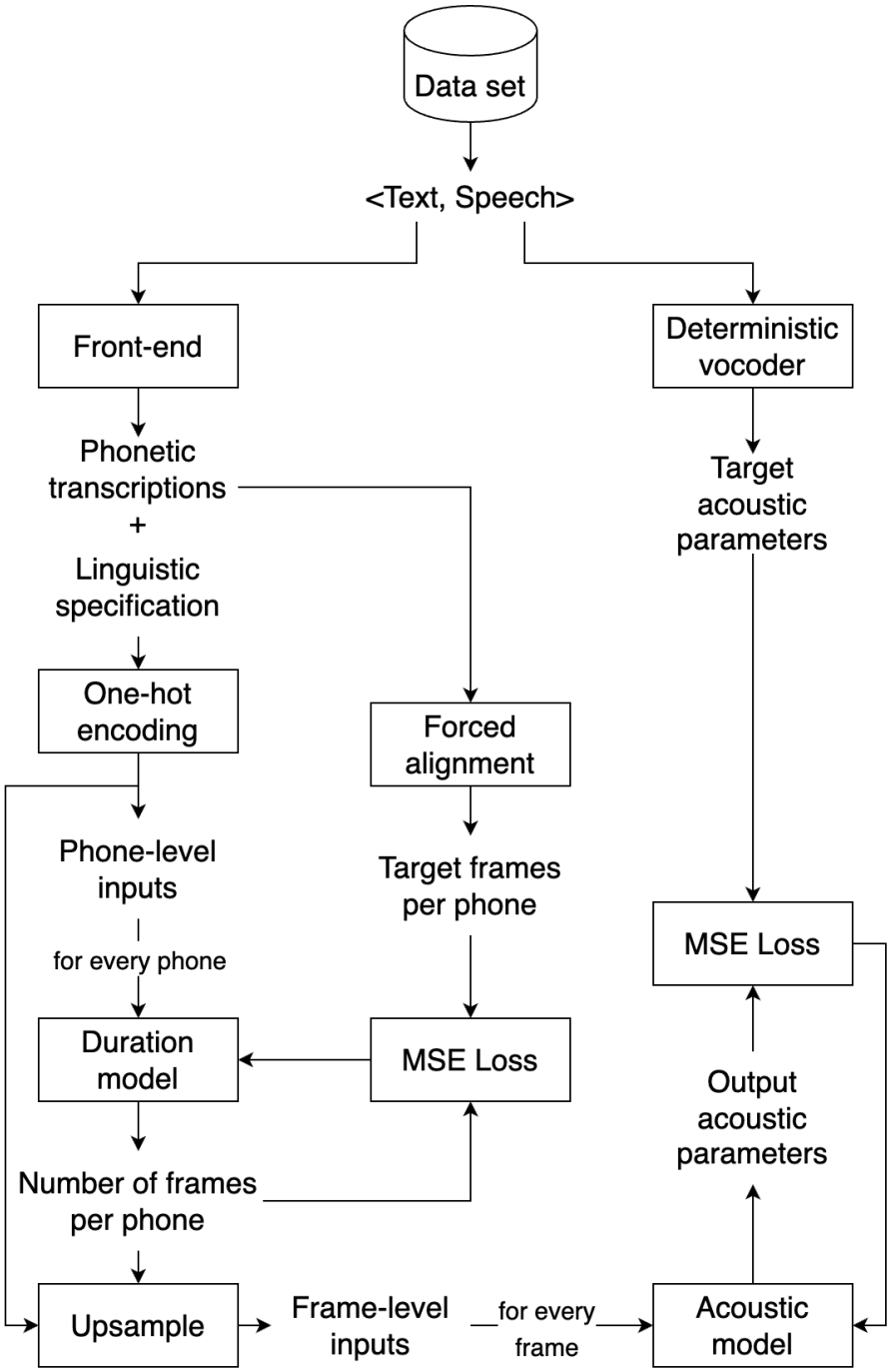

Figure 2.3: Diagram of the training procedure for frame-level neural network-based TTS using a data set composed of matching pairs of text and speech.

input is one frame at a time. Correspondingly, the target for the acoustic model is the frame acoustic feature. Spectral, fundamental frequency and aperiodic features are stacked into one vector, including voicing flag. Training, as well as before, can be performed using Stochastic Gradient Descent to minimize the MSE loss. As well as described for HMM-GMM-based models, post-processing trajectory smooth algorithms are applied when every frame has been generated independently. Finally, the generated acoustic vectors are run through the non-neural vocoder to obtain a waveform.

2.4.1.3 Architectures

The earlier neural network architectures used for TTS were purely based on feed-forward linear layers. For example, the architecture proposed by Zen et al. (2013) had four hidden layers with Sigmoid activations. The input layer had the dimension of the linguistic specification features: 371. The output layer generate means and variances of the acoustic features rather than features itself.

As we have emphasized in this section, in these neural networks, speech modelling happens frame by frame, independently. This is a disadvantage that applies also to HMM-GMM-based TTS systems too, because natural speech segments (i.e. vowels and consonants) are definitely dependent on each other. One approach proposed to improve dependency modelling across frames was to predict sets of four frames, rather than one at a time (Zen, Agiomyrgiannakis, Egberts, Henderson, & Szczepaniak, 2016). However, the most widely adopted approach, was the use of recurrent layers instead of, or together with, feed-forward ones (Zen, 2015). As described in Section 2.3.3, recurrent layers can model long-term dependencies through time, by feeding-back the hidden state for, in this case, previous frames. These layers can be improved further with LSTM units, ReLu activations, and/or bidirectional layers.

For example, Zen and Sak (2015) trained a duration and an acoustic model with one LSTM-recurrent hidden layer and one recurrent output layer. The results showed that the use of the hidden LSTM layer significantly improves the naturalness of the synthetic speech, and that the output recurrent layer has an effect equivalent to trajectory smoothing algorithms such as MLPG (as in Section 2.2.2).

Although these new architectures improved naturalness by better modelling dependency between frames, the generated synthetic speech suffered similar problems to HMM-GMM-based models, such as lack of expressivity, and audio quality characteristic of the use of classical vocoders. These issues could only be further improved through the adoption of sequence-to-sequence architectures which are reviewed in detail in the next section.

2.4.2 Sequence-to-sequence-based Text-to-Speech

Sequence-to-sequence (S2S) architectures are the state-of-the-art for TTS (and for many other deep learning fields) at the time of this thesis. In contrast to the architectures reviewed in the previous section, which model speech frame by frame, these architectures attempt to model the complete input sequence in relation to the output sequence. For sequence-to-sequence models, both the input and the targets are sequences defined by a length and a feature dimension.

The introduction of these models into TTS was heavily influenced by developments in the field of neural machine translation (NMT). Both machine translation

and TTS are tasks that map sequences. In the case of NMT, the input sequence is a sentence from the source language that is mapped to a target sequence corresponding to a sentence from the target language. In the case of TTS, the input sequence is a sequence of linguistic representations for each phone of an utterance's phonetic transcription that is mapped to a target sequence that is a sequence of frame-level acoustic representations across time.

In both cases, if we tried to map these two sequences directly, we would run into the same problem: the two sequences most likely will not have the same length, and even if they do, there is not necessarily a one-to-one correspondence between elements of each sequence at the same position. In NMT, a group of words in the input sequence might translate into only one word in the output sequence, or vice versa, or words might not have a counterpart at all in the other sequence. In TTS, one phone in the input sequence will most likely correspond to multiple frame-level acoustic frames in the output sequence, such that the output sequence will be about 4 times longer than the input phone-level sequence for typical acoustic frame rates.

In this section, we revise the different approaches that aimed at solving this fundamental issue, and that resulted in the design of sequence-to-sequence (S2S) architectures. In Section 2.4.2.1 we define encoder-decoder models, which were the starting point. Then, encoder-decoder architectures were further enhanced with the use of attention (Section 2.4.2.2). Both of these approaches are used in TTS architectures like Tacotron 2 (Section 2.4.2.3). The latest version of sequence-to-sequence models are transformers (Section 2.4.2.4), which exploit self-attention and multi-head attention (Section 2.4.2.5). These are the elements on which architectures like FastPitch are based (Section 2.4.2.7). All experiments in this thesis have been carried on using one of these two architectures, therefore, they will be described in detail.

2.4.2.1 Encoder-decoder architectures

Encode-decoder models proposed for NMT were a solution to the sequence-to-sequence mapping issue described in the section above (Stahlberg, 2020). The encoder is a neural network that summarizes the complete input sequence into a fixed-length vector. Then, the decoder is another neural network that consumes this vector to map it to the target sequence. Usually, the decoder will be auto-regressive, generating a new step conditioning on the previous hidden state too (Stahlberg, 2020).

Although these models showed improvements with respect to previous frameworks for machine translation, the fixed-length vector produced by the encoder lacks representational power, especially for longer sequences (Stahlberg, 2020). While this aspect of the architecture was further improved (as we will see below), the encoder-decoder design for architectures persists.

2.4.2.2 Aligning through attention

Recall how at the start of Section 2.4.2 we explained how the lack of a one-to-one relationship between the elements of the input and target sequences in TTS and NMT makes it difficult to map them directly. This can be formulated in terms of a hidden alignment between the sequences that needs to be found. Bahdanau et al. (2015) proposed an alternative approach to this problem for NMT: attention. An

attention mechanism is trained to provide scores for how relevant every element of the input sequence is to each element in another sequence.

There are many types of attention mechanisms, designed for different purposes. In this section we introduce attention as a method for aligning the input and target sequences for an encoder-decoder architecture. Later, in Section 2.4.2.5 we describe how attention can be used for feature extraction. See Figure 2.4 to follow the description below.

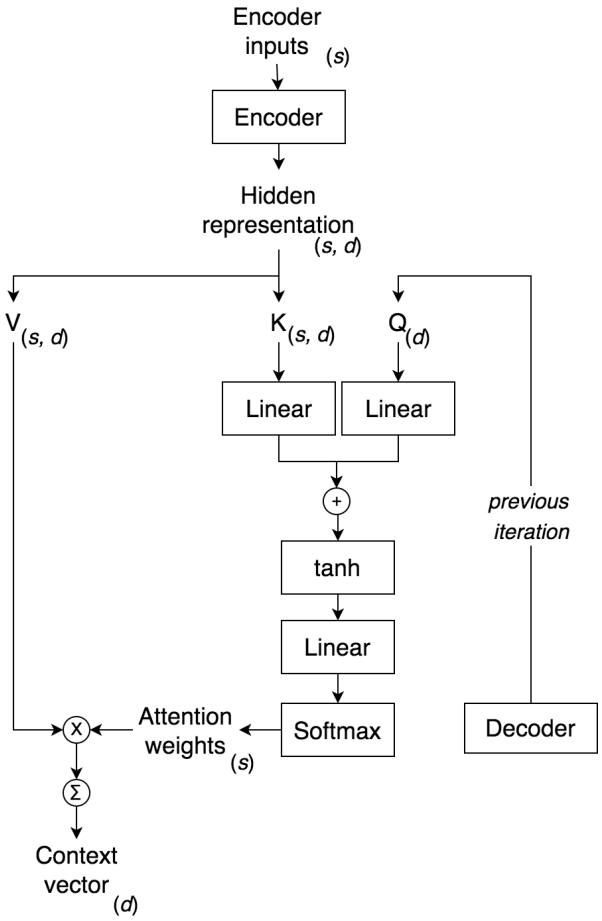

Figure 2.4: Bahdanau et al. (2015) additive attention. We use lower case symbols for the vector dimensions (s and d) to contrast them to the matrices K, Q and V.

Suppose that we have an encoder that generates a hidden representation of length s with a feature dimension d, where s is also the length of the input sequence. We also have an auto-regressive recurrent decoder that needs to generate, iteratively, a sequence of the same length of the output. Therefore, for every step the decoder is run, we use the attention to determine which elements in the encoder hidden representation (encoder outputs) are relevant for the current decoder timestep. In general, attention mechanisms are based on three matrices (for a discussion of the "biological" inspiration for these terms see A. Zhang et al. (2021, p. 393)): query (Q), key (K) and value (V). In Bahdanau et al. (2015), Q is the output of the decoder's previous timestep (one element only, not a sequence). Generally, K and V are both the complete sequence of encoder outputs (and therefore, have a sequence length). Two versions are kept because K will be used to calculate the distance

to Q, and according to the results, V will be "filtered" (multiplied by the softmax output, known as the attention weights) to keep only the relevant elements.

As part of the attention mechanism a distance or scoring function must be defined to compare Q and K. Bahdanau et al. (2015) uses "additive attention": Q and K are input into independent linear layers that calculate intermediate features for each matrix. The output is summed, passed through the tanh activation function and then through a final layer that flattens the features dimension to 1 to output a vector of length s (hidden representation length). Softmax is applied to this vector, obtaining attention weights that range from 0 to 1 for every element in V. The resulting matrix is summed through time to obtain a final fixed-length vector. This vector is fed into the decoder during the current iteration to predict the next element in the target sequence.

The attention mechanism in the model proposed by Bahdanau et al. (2015) is learning an implicit alignment between input and output sequences for the NMT model. Later this was also applied directly to TTS architectures: forced alignment was replaced with an implicit alignment learnt during training, and, instead of training a duration model independently from an acoustic model, both are integrated in a single architecture. Tacotron 2 (Shen et al., 2018) was one of the first architectures with a encoder-decoder design with an alignment attention mechanism for TTS.

2.4.2.3 Tacotron 2

Tacotron 2 (Shen et al., 2018) is an attention-based sequence-to-sequence model that predicts mel spectrogram acoustic representations from input linguistic symbols. The architecture is an encoder-decoder model with an auto-regressive decoder. For a detailed description of the mel spectrogram see Sections 2.4.3 and 3.1. Attention performs an implicit alignment by calculating the relevance of all encoder outputs with respect to each mel spectrogram frame, through the auto-regressive decoder. Therefore, duration and acoustic modelling occur jointly within one model. Crucially, this means that because the alignment is implicitly inferred through the attention module, explicit timestamps obtained through forced alignment are no longer required (unlike for models with an explicit duration model, as we will see in Section 2.4.2.6).

Unfortunately, the original Tacotron 2 authors did not release the code for their model and the paper lacks the necessary detail for a reliable implementation. This is why in this thesis we use NVIDIA's[1], which, as stated in the acknowledgements section of the repository, has been informed by the original Tacotron 2 authors in private communication. The architecture details that we provide in this section are based on that implementation. Figure 2.5 shows an overview of Tacotron 2 main modules during training. Notice that some modules of Tacotron 2 perform operations for entire sequences (the pre-net, the encoder and the post-net). In contrast, the decoder (with an attention), works iteratively for each frame in the target mel spectrogram (shown in Figure 2.5 inside the dashed box), learning to predict t. The attention output dimension (D), is a hyperparameter to be chosen later. In the following we will describe in detail each of these modules.

[1] `https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/SpeechSynthesis/Tacotron2`

Encoder: the input to the encoder is the input sequence with length L, which can be either graphemes or phones. The sequence is embedded through a layer of dimension 512. The encoder consists of a cascade of three 1D convolutional layers, with filters that span 5 input symbols at the time. A final bi-directional LSTM layer in applied to obtain the encoder output sequence.

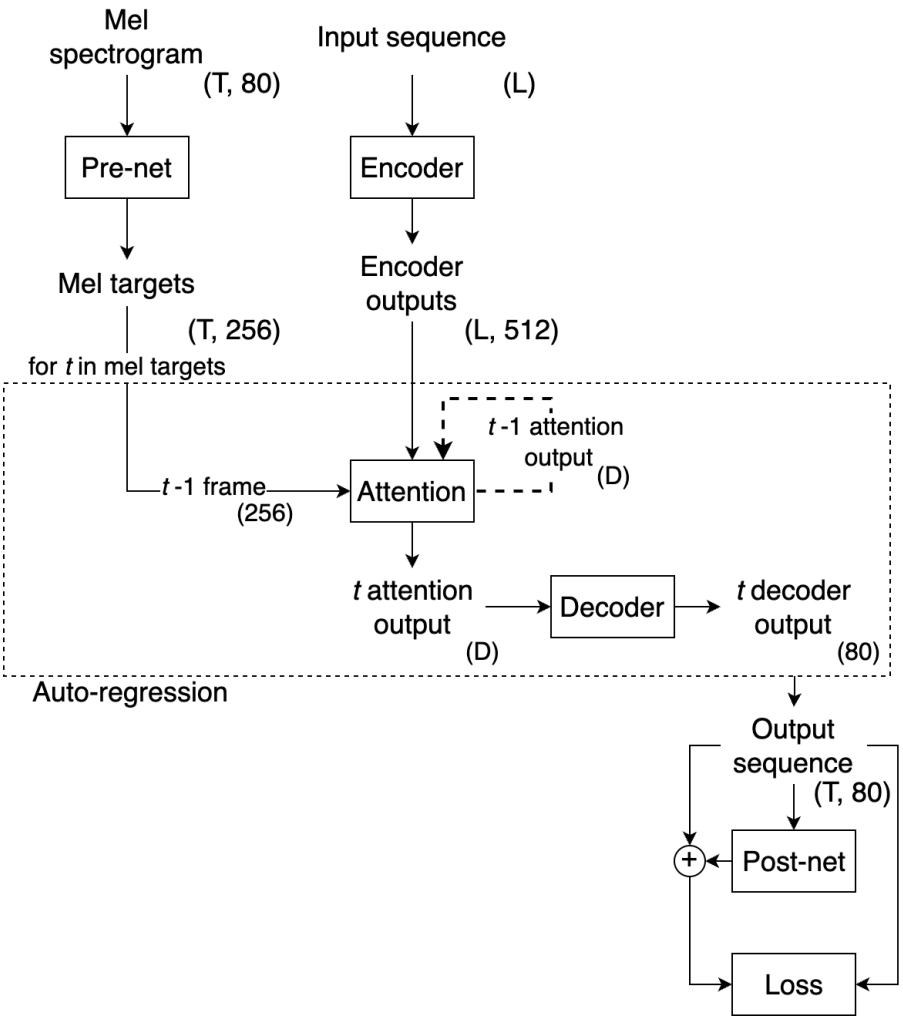

Figure 2.5: Tacotron 2 during training. We use T to indicate the number of mel spectrogram frames to emphasize the auto-regressive mode of the decoder.

Pre-net: in parallel, the pre-net is a feed-forward linear layer that pre-process the mel spectrogram. During training, when using teacher forcing (i.e. using ground-truth mel spectrograms), the target mel spectrogram sequence of length T is processed by this layer. During inference, in contrast, the input to the pre-net is the mel spectrogram frame previously synthesised by the auto-regressive decoder for the current utterance. During training, an initial zero-valued frame is concatenated to the beginning of the target mel spectrogram, such that at synthesis time, this frame can be used to initiate the auto-regression. According to the authors, the pre-net plays an essential role in order to ensure that attention is learnt.

Attention: the role of attention in Tacotron 2 is to calculate which steps in the encoder output sequence are relevant for the decoder in order to predict the next mel spectrogram frame. Figure 2.6 shows a diagram of the attention in detail.

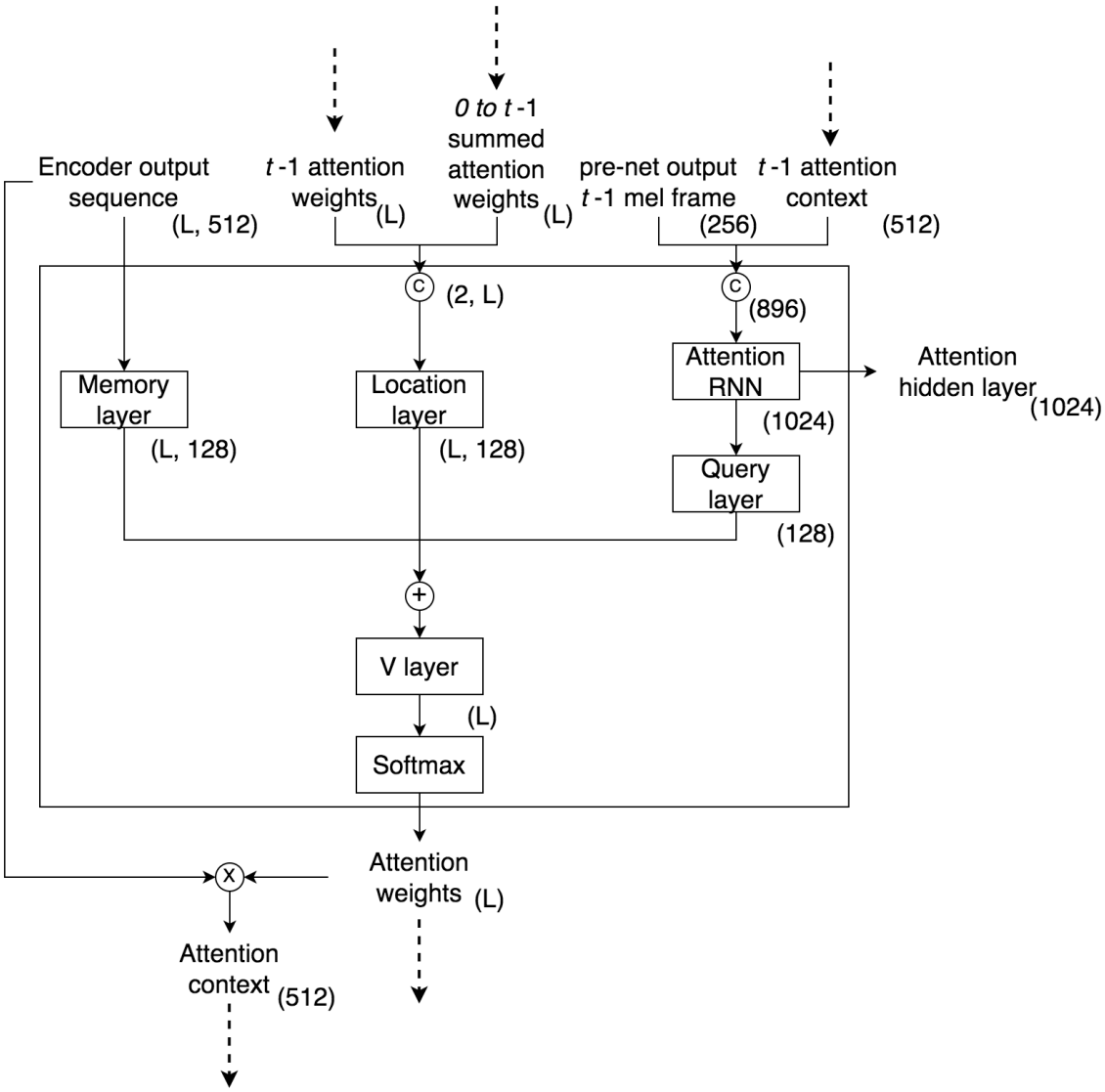

Figure 2.6: Tacotron 2 attention.

The attention module is the most complex part of the Tacotron 2 architecture, as many sources of information are used as inputs. The dashed arrows coming into the diagram at the top, show inputs to the attention module that are obtained through the previous time step computation. The dashed arrows going out of the diagram, at the bottom, show the same information generated in the current iteration that are fed forward in time. Because previous attention weights and context vectors are input into the attention module, this attention mechanism is called "location sensitive", as it is expected that such a design will encourage the attention alignment to move forward in time without requiring explicit monotonicity. The inputs to the attention are, following Figure 2.6 from left to right:

- Encoder output sequence: the complete sequence of encoder outputs with length L. Using the naming convention introduced in Section 2.4.2.2, the encoder output sequence that is processed by the memory layer corresponds to the K matrix, while the version that is directly passed to be multiplied by the attention context corresponds to the V matrix. This is the only input that remains constant for every t (and therefore in the implementation the memory layer is only run once).

- t-1 attention weights: the previous attention weights calculated at the previous time step that, in the way explained in Section 2.4.2.2, have the same length as K and V.

- 0 to t-1 summed attention weights: the accumulated sum of the attention weights calculated for every previous time step.

- the t-1 pre-net output frame from which that is expected to be continuity in order to predict the target mel frame t. This would be the conventional Q matrix as defined in Section 2.4.2.2, but as the diagram shows, all the other elements are also taken into account to calculate the relative importance of the elements in K and V.

- t-1 attention context: attention context vector calculated in the previous time step.

The outputs of the attention module are:

- Attention context: a vector that summarizes the relevance of the whole encoder output sequence for the current decoder time step with respect to all the inputs to the attention. This vector is fed-back to the attention recurrent layer. This is one of the attention outputs that is directly consumed by the decoder.

- Attention weights: of length L and containing the information of how relevant each step in the encoder output sequence is to the current decoder time step. These are accumulated and input into the attention. This is commonly plotted to inspect the "attention alignments".

- Attention hidden layer: extracted from the hidden representation obtained by the recurrent layer in the attention module. This is the only other attention output that is directly consumed by the decoder.

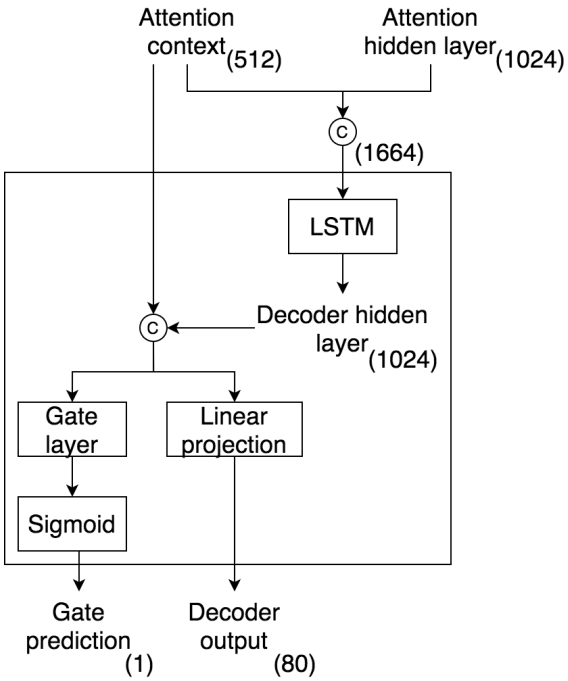

Figure 2.7: Tacotron 2 decoder.

Decoder: the attention module generates all the pieces of information that go into the decoder. Notice that the mel spectrogram never goes directly into the decoder. None of the representations in the decoder are sequences, as all of them correspond to only one time step. The attention context is a core piece of information that is concatenated twice inside the decoder, before and after the LSTM layer. The concatenated vectors are used to predict the decoder output, the mel spectrogram frame t, and a gate prediction, which is binary vector signaling if the utterance has finished or not. See Figure 2.7 for details.

Post-net: (refer back to Figure 2.5) once all the decoder outputs have been accumulated into the output sequence of length T, the final sequence is passed through a post-net, which consists of five 1D convolutional layers. The role of this module is to improve the predictions once all frames in the sequence are present, predicting a residual that is added back to the decoder outputs, similar to the smoothing operation described for HMM and frame-level neural network-based TTS in previous sections.

Loss: three terms are summed to obtain the final loss: (1) an MSE loss over the decoder outputs, (2) an MSE loss over the post-net outputs, and (3) a binary cross entropy loss over the stop token (gate) prediction.

Batch normalization, ReLU and drop-out (or zone-out, as in the original paper) are applied throughout the network in the same locations as in the original architecture but these are omitted from our figures for the sake of simplicity.

A major drawback of Tacotron 2 for quick experimentation is slow training time of the auto-regressive decoder. In Section 2.4.2.6 we will discuss further disadvantages of using attention as an implicit duration model. Overall, the naturalness of the resulting synthetic speech generated with this model can be very high, and it is still a state-of-the-art architecture widely used. In this thesis we make use of this architecture for the experiments in Chapter 4 and Chapter 6.

2.4.2.4 Transformers

After the good results obtained with sequence-to-sequence models with attention, research increasingly focused on how to further improve attention mechanisms. Vaswani et al. (2017) introduced a new architecture called transformers which is solely based on stacking multiple attention and linear layers for both the encoder and the decoder. These models were, once again, developed first for NMT, and were found superior to the encoder-attention-decoder architecture explained in Section 2.4.2.2 and, crucially, significantly faster. The speed improvements come from discarding auto-regression and the need for recurrent or convolutional layers. This architecture had particularly relevant consequences in NLP, as transformed-based large pre-trained language models (Devlin, Chang, Lee, & Toutanova, 2018) are one of the most successful proposed in the field (Section 2.6.1). Transformers are also popular in recently proposed TTS architectures such as FastPitch (Łańcucki, 2021), which we will review in Section 2.4.2.7.

The transformer of Vaswani et al. (2017) is still designed with an encoder and a decoder, but a new attention mechanism called self-attention, together with multi-head attention, are at the core of the architecture (see next section). The input to the encoder is a sequence, which is first embedded. The encoder is composed of a stack of 6 modules, each one containing a multi-head self-attention mechanism, followed by 2 linear layers and residual connections which are applied throughout the encoder. Unlike previous auto-regressive sequence-to-sequence models, where every time step generates a single output element, the new attention mechanisms are applied to the full sequence without positional knowledge. To provide this, a positional encoding is applied, to assign a vector to every element in the attention input that indicates its position in the sequence. The decoder has a design that mirrors the encoder, such that the target sequence is also embedded and passed to the positional encoding, but shifted by one element to the right. It is composed of a stack of 6 modules, but each module has two multi-head self-attention layers: the first one processes the target sequence. The second one, connected to the output of the first one, processes the encoder outputs. Because this model was designed for NMT, the two final feed-forward layers are connected to a final layer and softmax to produce probabilities over words.

2.4.2.5 Self-attention and multi-head attention

Self-attention, as the name suggests, is applied to a single sequence only, where the attention mechanism scores the relevance of its items with respect to themselves: K, Q and V all correspond to the same sequence. In transformers, self-attention is applied both in the encoder and the decoder. All self-attention layers in the encoder modules are identical. In the decoder modules, the self-attention that processes the target sequence, is masked such that no element can attend to others that are relatively positioned in the future. Then, the attention layer that takes the output of the encoder, behaves similarly to the attention described in Section 2.4.2.2, learning the relationship between the encoder output elements and the processed target sequence. According to Vaswani et al. (2017), self-attention not only helps to reduce the computation and complexity of the model, but also improves the learning of long sequences. As a scoring function, Vaswani et al. (2017) uses scaled-dot product attention. K, Q and V are passed through independent linear layers.

For the output Q and K, the dot product is calculated, and divided by a scaling factor (which helps with vanishing gradients). Then, softmax is applied and the resulting scores multiplied by V to obtain a weighted version of this matrix.

Additionally to the self-attention, a multi-head attention mechanism is implemented. The idea is that by calculating the same attention several times, different patterns can be found. Instead of running the same process multiple times, the K, Q and V matrices are projected into a dimension that is later split given the selected number of heads. It is assumed that each of these heads is encoding a different pattern. The matrices are split and the scoring function is applied to every matching slice of K and Q, and correspondingly, the attention weights applied to filter the slice of V for the given head. All the results are concatenated and finally projected one last time to obtain the final output. For a more detailed description of the multi-head mechanism, see Section 2.6.3.

2.4.2.6 Explicit duration modelling

As mentioned in Section 2.4.2.3, although the use of attention as an implicit duration model for architectures such as Tacotron 2 increased synthetic speech naturalness, it also brought some disadvantages. For example, learning the attention weights was not always guaranteed. A solution proposed to cope with this issue can be found in Zhu et al. (2019), where the attention weights are initialized from timestamps obtained through forced alignment. In this thesis, when training Tacotron 2, to handle this issue, we had to train for a few epochs first while applying a reduction factor to the mel spectrogram (where we remove every other frame, reducing the length by a factor of 2) in order to obtain an initial attention alignment. After a few epochs, and inspecting that the attention alignments are approximately diagonal, we would switch to training with the full mel spectrogram.

Even when the attention weights are successfully learnt, it is still possible to find errors caused by attention during synthesis. In Fong et al. (2019) we investigated how when training systems with mismatches in the training data between the input symbol sequence and the acoustic target, these systems can present attention failures when synthesizing, resulting, for example, in skipped words or causing early stopping. Some models, such as DC-TTS (Tachibana, Uenoyama, & Aihara, 2018) enforce the attention weights to be nearly diagonal, by allowing attention over a small sliding window of the encoder outputs only. This prevents some of the failures mentioned before from happening.

Considering the issues found when using attention as an implicit duration model for TTS, subsequent architectures discarded it and returned to using explicit duration models. In contrast with the duration models used in HMM-GMM-based and frame-level neural network-based speech synthesis, these duration models are trained jointly with the acoustic model. The main disadvantage of this approach is that, once again, we need timestamps for the training data to be used as targets for the duration model. FastPitch (Łańcucki, 2021) is one such architecture, which in addition to incorporating an explicit duration model uses transformers for the encoder and decoder.

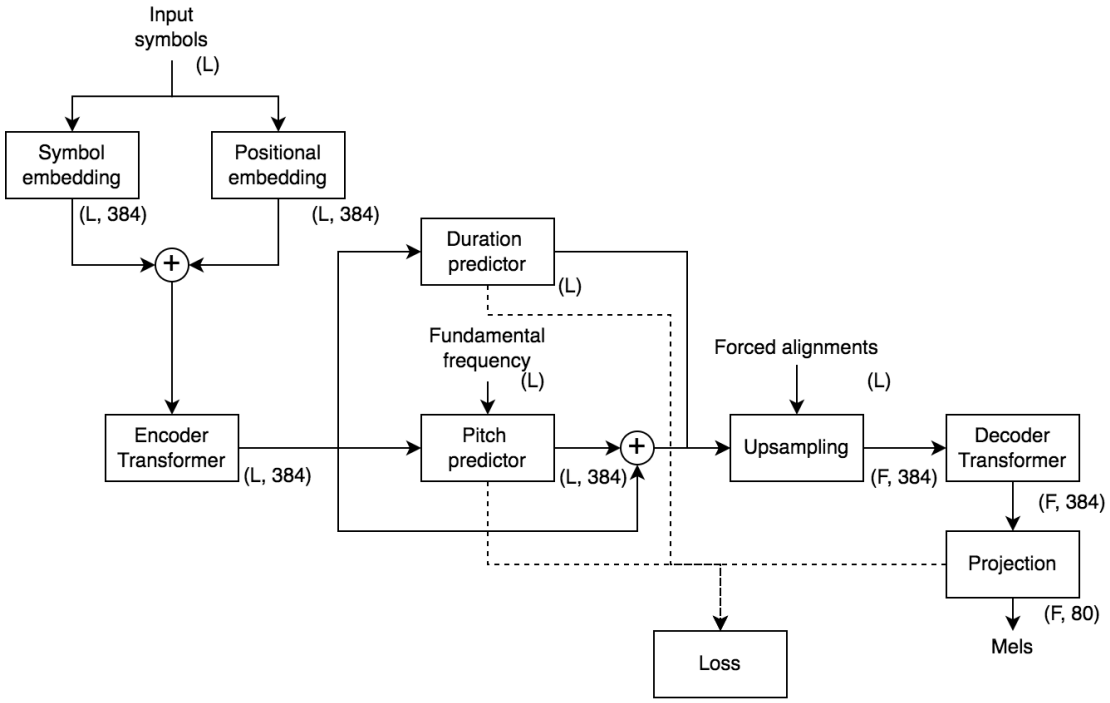

Figure 2.8: FastPitch overall architecture. We prefer to use F to indicate the number of mel spectrogram frames to emphasize upsampling (in contrast to Tacotron 2's auto-regression).

2.4.2.7 FastPitch

FastPitch (Łańcucki, 2021) is a transformer-based encoder-decoder TTS system with explicit duration and fundamental frequency predictors. Figure 2.8 shows the overall architecture. While the encoder processes sequences of phonemes (of length L), the decoder operates over the sequence of frames (of length F). To connect these, the encoder output must be upsampled to match the number of frames in the decoder target sequence.

As explained before, through the use of transformers and by discarding auto-regression, FastPitch is considerably faster to train while still achieving synthetic speech quality as good as Tacotron 2. This is the base architecture that we use for many of the experiments in this thesis. In the remainder of this section we describe in detail each part of the architecture, based both on the paper (Łańcucki, 2021) and on the implementation provided by the author[2]:

Input embeddings and encoder: the input to the encoder is the sum of the embedding of the input symbols (phones) and the positional embeddings. Figure 2.9 shows a detailed diagram of the transformer encoder architecture. The transformer encoder is composed by a stack of 6 transformer layers, integrated by a multi-head self-attention (with 1 head) module (blue block in the figure) and followed by a block of 1D convolutions (green block in the figure), both including residual connections. As the figure shows (dashed arrow), the output of each stacked transformer layer is fed into the next one.

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

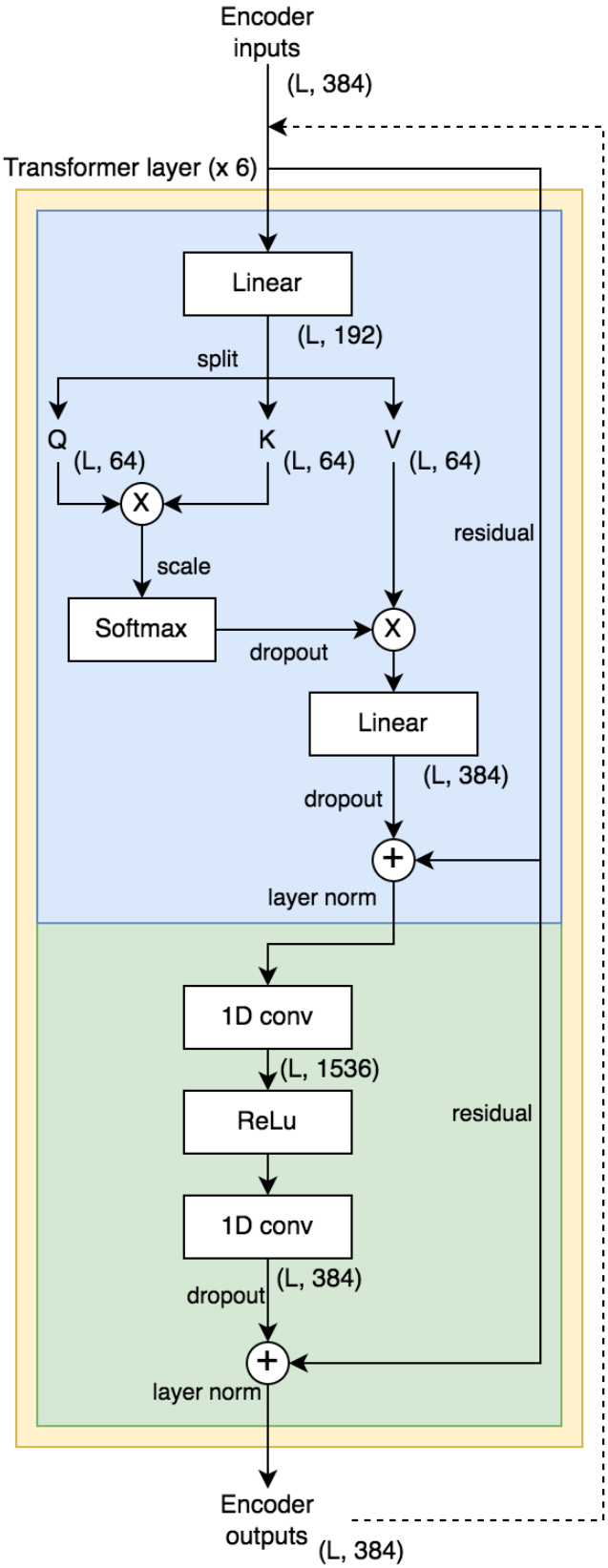

Figure 2.9: FastPitch encoder transformer.

Pitch (f0) prediction: the pitch predictor has two parts: the first is a block of convolutions, similar to the ones in the transformer encoder, followed by a linear layer, and a second part, which is an embedding layer. During training, the convolutional block takes as input the encoder outputs, while the embedding layer takes the ground-truth normalized fundamental frequency contour (teacher forcing). During synthesis, the convolution block predicts the fundamental frequency contour from the encoder outputs instead, which is processed by the embedding layer. This block has two sets of 1D convolutional layers followed by ReLU activations, layer norm and dropout. The first convolution keeps the dimensionality at 384, the second reduces it to 256 and the final linear layer flattens the output to 1. The embedding takes the dimensionality back to 384, and the output is summed to the encoder outputs. Notice that the fundamental frequency is predicted at the phone-level (L). Therefore, as part of the data pre-processing, fundamental frequency contours are extracted for every utterance. Using the forced alignment timestamps, the fundamental frequency is averaged for every phone, obtaining a vector of length L. The fundamental frequency values are normalized to zero mean and variance of one.

Duration prediction and upsampling: the duration predictor has the same architecture as the pitch predictor, except for the final embedding layer. This predictor also takes as input the encoder outputs, and predicts log duration. During training, the sum of the pitch predictor and the encoder outputs is upsampled from L to F according to the forced alignment timestamps (teacher forcing). During synthesis, the output of the duration predictor is used instead. Note that before the upsampling all the matrices are of length L while after it all matrices are of length F, matching the length of the target sequence: the mel spectrogram.

Decoder and projection: the decoder has the same architecture as the encoder as shown in Figure 2.9, with the exception that L is replaced with F. A final linear projection layer is used to re-dimension the decoder outputs to the 80-dim mel spectrogram frames.

Loss: the loss of FastPitch has three terms, all of them MSE: (1) pitch, (2) log duration and (3) mel spectrogram, all equally weighted. The outputs of the modules that contribute to the loss are indicated in Figure 2.9 with dashed arrows.

In this thesis, all experiments are carried on using either Tacotron 2 or FastPitch. As mentioned before, Łańcucki (2021) showed through a listening test that FastPitch achieves comparable synthetic speech quality to Tacotron 2. FastPitch, as well as being faster than Tacotron 2, is more robust to different kinds of data. Because FastPitch does not rely on attention during training to find the alignment between sequences, we do not have issues to achieve training convergence. Consequently, the failures seen for Tacotron 2 at synthesis time originating from attention are not seen for FastPitch. When there are errors in the synthesis with this model, these tend to be localized, for example, to a particular syllable or phone, unlike with Tacotron 2, for which when the attention fails, very commonly, the complete utterance is affected. Moreover, FastPitch jointly learns to predict the mel spectrogram, fundamental frequency and duration. This makes the model more supervised than Tacotron 2, and together with the improved transformer architecture, make it an even stronger baseline for us to beat.

2.4.3 Neural vocoders

As mentioned at the start of Section 2.4, one of the innovations that allowed increases in overall quality of synthetic speech together with sequence-to-sequence models was the introduction of neural vocoders. These vocoders, unlike the classical ones used with previous TTS frameworks, are neural network-based trainable models. As we explained in Section 2.2.2, older vocoders were characterised by signal processing artifacts that decrease the generated speech quality. Modern neural vocoders inherently do not generate speech with such characteristics.

These neural networks are trained to map from the intermediate acoustic features, generated by the TTS architecture, to waveforms. Although some neural vocoders work for other types of features, most of them take as input a mel spectrogram. Mel spectrograms are obtained following the same steps described in Section 2.2.3 to obtain MGCs, but instead of applying a frequency warping transform, a mel-filter bank is applied. Also, there is no need to decorrelate the features this time. To go back to the waveform, the neural vocoder needs to reconstruct all the information that was lost during the mel spectrogram extraction: the frequencies lost by the mel-bank filters, (especially for higher frequencies) and the discarded phase (Govalkar, Fischer, Zalkow, & Dittmar, 2019). In Section 3.1 for details on how mel spectrograms are extracted for our experiments.

As an alternative to neural vocoders, some experiments make use of the Griffin-Lim (GL) algorithm (Griffin & Lim, 1984). In this case, first, the magnitude spectrogram is mathematically approximated from the mel spectrogram. Then, the inverse Short-Term Fourier Transform is applied and GL is used to reconstruct the phase to obtain the final waveform. Although this is a fast method compared to neural vocoders, and so can be used for quick experimentation, the quality of the resulting speech is usually not comparable to that from a neural vocoder.

Multiple neural vocoder architectures have been proposed. For an example comparison, Govalkar et al. (2019) can be consulted. However, in this thesis, we only make use of WaveGlow (Prenger, Valle, & Catanzaro, 2019). As mentioned before, this thesis does not focus on improving the neural vocoder: we use it as a black-box. Nevertheless, we give some details of the implementation that we use in this thesis in the next section.

2.4.3.1 WaveGlow

WaveGlow (Prenger et al., 2019) is a non-autoregressive, generative, flow-based model that efficiently generates high quality waveforms from mel spectrograms. Crucially, flow-based models, instead of having an encoder and a decoder, are composed of invertible transformations, such that the mapping can be done back and forth. When training the model, the samples from the input waveform are grouped into arrays of eight samples. These are processed by 12 blocks of invertible 1 x 1 convolutions and an affine coupling layer that outputs the samples distribution. The coupling layers make use of convolutional layers similar to the ones proposed in WaveNet (Van den Oord et al., 2016). At this point, the model is conditioned on the mel spectrogram. The result of the final block is the final distribution, which, during inference, is randomly sampled. The result is fed-back into the network, which is also conditioned on the mel spectrogram. The network, this time is inverted, generating the corresponding waveform samples.

We chose this vocoder because, besides its high quality and quick inference time, has openly released code[3] and pre-trained models[4]. Moreover, the code source for Tacotron 2 and FastPitch that we use as a starting point for our experiments was released by the same company, and so, it is very easy to integrate. We use the default implementation available in the repository. By using pre-trained models, we avoid having to train them ourselves. Moreover, the pre-trained checkpoint we make use of has been trained with LJ Speech (see Section 3.2), a data set that we also use for TTS. Although we could have fine-tuned this checkpoint for other data sets that we use in this thesis, we did not, either because the checkpoint worked well for them or because fine-tuning did not bring improvements. Finally, because the vocoder is not the focus of our experiments, we only make sure that, when evaluating systems, all synthetic speech has been generated using the same checkpoint to avoid a confound.

2.5 Evaluation

When a new TTS architecture or an improvement over an existing one is proposed, testing must be done to show that the newly generated synthetic speech is significantly better with respect to a baseline. The baseline can be the current state-of-the-art, or the base architecture for which a method seeking improvement has been proposed. Usually, natural speech is also considered within the test comparisons as the ground-truth.

There are two general speech characteristics that are commonly evaluated: intelligibility and naturalness (Taylor, 2009, p. 535). Although there is not an agreed definition of these concepts, from a phonetics perspective, intelligibility can be understood as segmental quality only, while naturalness is related to both segmental and suprasegmental quality. HMM-GMM-based TTS systems were already capable of highly intelligible speech with as little as ten minutes of training data (Zen et al., 2009). Although naturalness has been dramatically improved in the last years by state-of-the-art architectures, improvements are still possible, and new research tends to focus on this rather than on intelligibility.

Both of these speech characteristics can be evaluated through objective or subjective methods (Wagner et al., 2019). Objective methods are based on automatic tests that usually provide a score or value for a specific speech dimension that approximates intelligibility or naturalness. For example, speech intelligibility can be approximated by running a speech-to-text system on synthetic speech (Baby et al., 2020). The obtained transcription is compared to the ground-truth, and a word error rate (or phone error rate) can be calculated. For naturalness, objective tests of fundamental frequency error, speech rate error, spectral distortion, intensity error, among others, can be used to approximate the naturalness.

In contrast, subjective evaluation, e.g. listening tests, consist of having a group of listeners that judges the synthetic speech and provides a score or a preference choice given the listening test instructions. Although objective evaluations are good for quick experimentation they are only approximations and there are still no objective evaluations capable of capturing the full extent of what listeners judge through listening tests. Recent research has proposed making use of machine learning tech-

[3] `https://github.com/NVIDIA/waveglow/`

[4] `https://catalog.ngc.nvidia.com/orgs/nvidia/models/waveglow_ljs_256channels`

niques to predict listening test scores or preferences, such as Lo et al. (2019), based on supervised training. These models have limited success (Williams, Rownicka, Oplustil-Gallegos, & King, 2020; Wagner et al., 2019).

Recently it has been pointed out how the evaluation of a particular sentence can be affected by the lack of context when the speech is evaluated. In natural speech, the exact same sentence could be considered natural in many different spoken renditions, and such judgement will dependent on context. But, most evaluation in TTS is applied utterance-by-utterance without considering contextual information beyond that. An increasing number of authors are advocating for the need to incorporate context when evaluating speech (Latorre, Yanagisawa, Wan, Kolluru, & Gales, 2014; Clark et al., 2019; Wagner et al., 2019), but because efforts are only starting, there is not yet a reliable in-context evaluation design.

Considering the lack of correlation between objective and subjective evaluations, in most of this thesis we evaluate through listening tests, referencing objective evaluations from time to time with the intention of quick model selection or for analysis. In Chapter 8, where we focus on evaluation, we explore some methods related to automatic evaluation. Considering that there is not yet an agreed in-context listening test design, in most of this thesis we perform evaluation for isolated utterances, except in Chapter 8 where we handle in-context evaluation in-depth.

As mentioned before, usually, in a listening test a group of listeners will be asked to listen to and judge a set of synthetic speech samples (sometimes including natural speech as a reference too). There are many different listening test designs. The selected design will depend on what is being experimented with, what hypothesis is being tested and the number of TTS systems that will be compared. When designing a listening test, multiple variables need to be considered and reported: what are the instructions that participants receive, how participants are selected and remunerated, how many participants will take the test, how many test sentences will be used, how will samples be presented to the listeners, what kind of answer is expected from the listeners (numerical, selection, comment), is the test in-person or online, among others (Wester, Valentini-Botinhao, & Henter, 2015). In the following sections we describe some of the most common listening test designs, which are also the ones we make use of in this thesis.

2.5.1 Multiple Stimulus test with Hidden Reference and Anchor

The MUltiple Stimulus test with Hidden Reference and Anchor (MUSHRA) (ITU, 2014) test was proposed first as a framework for the subjective evaluation of audio quality. In a typical MUSHRA test there is a reference at the top of the screen (see Figure 2.10) which in a TTS listening test would be, for example, the natural speech, to give an idea of the highest quality expected by any of the systems being rated. Below the reference, there is a set of samples with the same linguistic content as the reference, each one generated by one of the TTS systems we are evaluating, and additionally the same reference presented at the top, but hidden among the options. The original MUSHRA test also includes an anchor which is supposed to be a reference of the lowest quality expected, but as it is not clear what that would be in synthetic speech, it is usually not included (and that is why in this thesis, when using this design, we refer to it as a *MUSHRA-like* design).

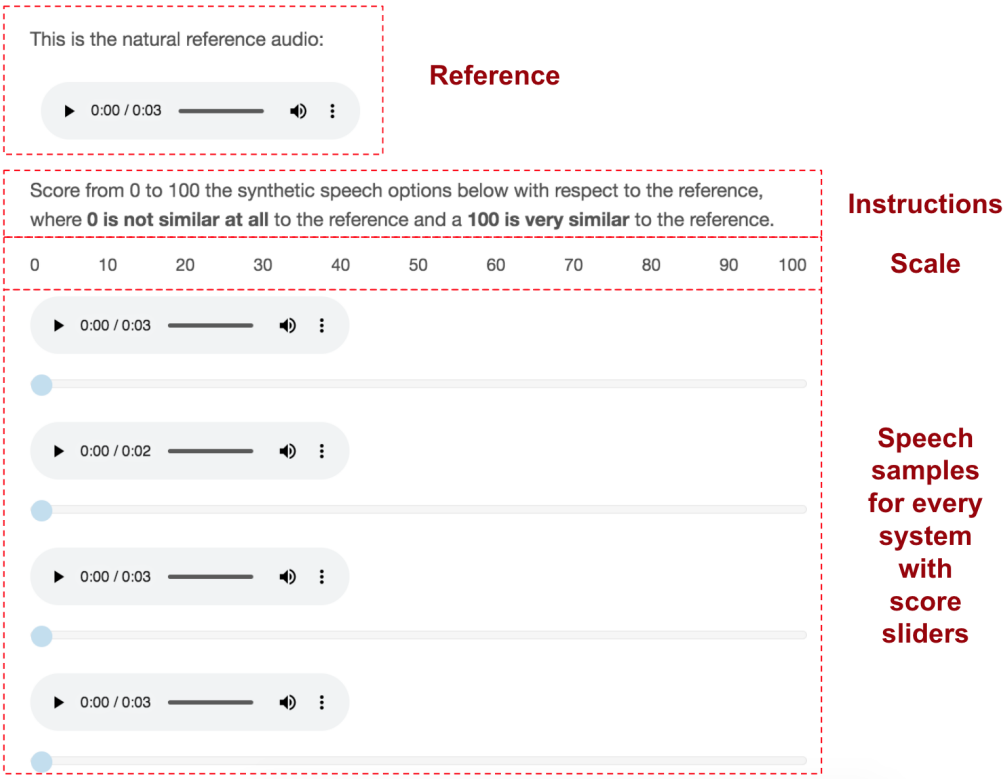

Figure 2.10: Example of a screen from a MUSHRA-like listening test for TTS, showing the elements of the design.

Each sample from each system needs to be rated from 0 to 100. What these scores represent will depend on the instructions given to the participants: for example, if the listeners are asked to rate the naturalness of the speech, 0 can be "not natural at all" and a 100 "extremely natural". In Figure 2.10, the instruction is to rate the similarity of the samples to the reference. The participants are instructed to find the reference hidden among the options and give a score of 100 to it. The participant will listen to as many screens (the combination of reference and samples below it, like Figure 2.10) as there are samples in the test. It is common to also provide the text of the sample.

In this thesis we use MUSHRA-like listening tests when we compare more than two systems and when we believe that having a reference is important for the hypothesis we are testing with that experiment. Although the design allows for many systems, we usually do not compare more than six, to avoid participant fatigue. To determine if the resulting scores of a MUSHRA test differ significantly, we first test if the scores are normally distributed using the Shapiro-Wilk test. If they are, a t-test can be applied to determine which scores statistically significantly differ. If they are not, we use the Wilcoxon-signed rank test. Because usually we are making several comparisons in MUSHRA-like tests, a Bonferroni correction is applied to the p-value.

2.5.2 Mean Opinion Score

In a Mean Opinion Score (MOS) (ITU, 2006) evaluation there is usually no reference, unless stated otherwise. Each synthetic speech sample (or natural speech,

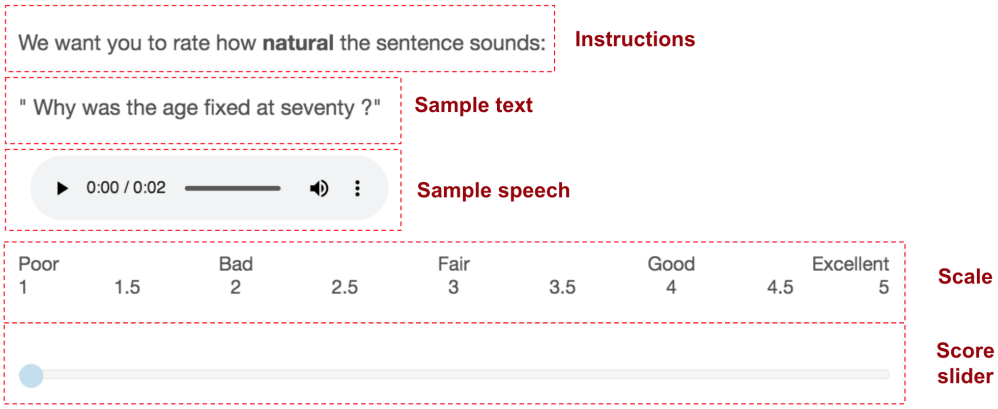

Figure 2.11: Example of an MOS listening test sample for TTS and the elements of the design.

if included) is rated separately (see Figure 2.11), unless stated otherwise (which is sometimes called side-by-side MOS). The rating scale varies, but most authors use a Likert-like scale, from 1 to 7 or 1 to 5. As for MUSHRA, these scores are interpreted depending on the instructions of the test, such that 1 could be "Poor" and 5 or 7, "Excellent". For a discussion on the use of MOS as a listening test framework for TTS we recommend Viswanathan and Viswanathan (2005). The text of the sample can also be presented if the test requires it, as in Figure 2.11. We can use the same statistical tests described for MUSHRA.

2.5.3 Preference test

When we are comparing only two systems in this thesis, a preference test, also known as an A/B test, is used. This listening test framework usually does not involve a reference, unless stated otherwise. Two samples are compared at a time. Commonly, the listener will be asked to indicate which of the two option is preferred given the instruction: for example, which is the most natural. When the participant *must* select one of the two options we refer to it as a forced choice listening test. In other designs, a "no preference" option is provided. The text indicating the linguistic content of the sample can be provided. Preference tests also tend to be used when the hypothesis predicts a clear preference between two systems. To determine if the resulting preference is statistically significant, we use the Binomial test.

2.6 Learnt representations

As mentioned in Section 2.1.3, when describing a standard front end for TTS and the input sources commonly used to train these systems, state-of-the-art architectures tend to be improved by incorporating learnt representations. Phonetic transcriptions, timestamps or fundamental frequency contours, are attributes extracted directly from the data. Even if learnt representations are learnt from the same data, they are more abstract and capture patterns that are hidden in it. In order to learn these representations, complex state-of-the-art architectures are leveraged.

For TTS, we will usually be interested in learnt representations either for text or for audio. To obtain such representations, we can either use very large pre-trained models or learn these representations jointly with the TTS architecture. There is also the in-between approach of fine-tuning a large pre-trained model to the TTS task. Throughout this thesis we make use of both approaches, and therefore in this section we describe them in detail.

In Section 2.6.1 we describe the Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al., 2018), the most popular large pre-trained language model at the moment. A language model is trained to predict the probability of word sequences given the training data. In the case of BERT, which has been trained with a massive data set of text, this model can be used to extract representations for text sentences that capture contextual patterns.

Similarly, we can extract acoustic representations for the speech. Many unsupervised models had been proposed to extract representations from audio in recent years, such as wav2vec (Schneider, Baevski, Collobert, & Auli, 2019; Baevski, Zhou, Mohamed, & Auli, 2020; Baevski, Schneider, & Auli, 2019), HuBERT (Hsu et al., 2021) and Mockingjay (Liu, Yang, Chi, Hsu, & Lee, 2020), among others. In this thesis, however, we use DeepSpectrum (Amiriparian et al., 2017), which in contrast to the approaches mentioned above, uses large pre-trained *image* classification models to extract representations from spectrogram plots. We chose this model for our experiments based on results obtained in collaborations parallel to this thesis (Williams et al., 2020; Oplustil-Gallegos, Williams, Rownicka, & King, 2020).

We also experiment with learning speech representations jointly with the TTS task. We use the Global Style Token method proposed by Y. Wang et al. (2018) in many experiments in this thesis, and therefore, we give a detailed explanation of how this method works in Section 2.6.3. Although this method was originally proposed to learn acoustic representations, we investigate how it can be generalized to other representations and also used in combination with large pre-trained model-extracted representations, in Chapter 5.

Once the acoustic or textual representations have been either extracted with large pre-trained models or with architectures jointly trained with the TTS system, it is common to condition the TTS system on them to obtain improvements. Typically, a fixed-length vector is obtained, and the TTS model is conditioned on it, by either summing it or concatenating it within the TTS architecture. Summing has the advantage of keeping the same dimensionality, while when concatenating, the dimensionality of the next layer needs to be adjusted to receive the new input size. Usually in state-of-the-art architectures, conditioning happens at the encoder input or output. Although usually these representations are extracted for the same utterance (the current utterance), in this thesis we focus on extracting these representations for the context utterance and then conditioning the learning of the current utterance on it. This is the core idea of the experiments that will be presented in Chapters 4 and 5.

2.6.1 BERT: Bidirectional Encoder Representations from Transformers

The Bidirectional Encoder Representations from Transformers model (Devlin et al., 2018), commonly known as BERT, is one of the most groundbreaking models in deep

learning in recent times. It is a large language model pre-trained with a massive data set (3,300 million words) in an unsupervised fashion to learn token- and sentence-level representations. Usually, other researchers that want to make use of this model will fine-tune it for a downstream task of interest or they will use BERT as it is to extract representations for text data.

Devlin et al. (2018) claim three significant contributions of BERT: (1) By incorporating bidirectional context between text tokens, the learnt representations are improved with respect to other state-of-the-art language models; (2) The model is easy to fine-tune by training only one layer further for a downstream task, with no need to fine-tune all the model's parameters, thus requiring little data; (3) It achieves state-of-the-art performance for both sentence-level and token-level downstream tasks, showing improvements even over task-specific architectures.

BERT is a multi-layer transformer based on that of Vaswani et al. (2017) (Section 2.4.2.4) that uses bidirectional self-attention, e.g. every token in the sequence can attend to any other token in the sequence, without enforcing attending to past tokens only. BERT is trained with pairs of sentences consecutive in the source text materials (when running inference only it is optional to use the two sentences, only one can be used too). The input to BERT is a concatenation of three sequences: (1) tokens obtained through WordPiece (Y. Wu et al., 2016) are embedded and used as input; (2) a "segment" vector, that indicates if a token belongs to the first or second sentence in the input; and (3) a positional embedding (Section 2.4.2.4).

WordPiece (Y. Wu et al., 2016) is a model for tokenization that automatically finds sub-word units in text. The use of sub-word units allows the model to represent rare words through their components while keeping the most frequent words as whole units. The aim is to obtain a token-level representation for the training data with a minimal vocabulary of units that represents the whole text, ensuring that at inference time there will be no sub-word units previously unseen by the model. For BERT, Devlin et al. (2018) use a vocabulary of 30,000 tokens. At the start of the first sentence token sequence a "CLS" (i.e. classification) token is concatenated. The hidden representation that will be learnt for this token will be later used as the sentence-level representation of the input. After the first sentence, a "SEP" (i.e. separation) token is appended, followed by the second sentence sequence of tokens. The complete sequence is then embedded using a trainable look-up table.

BERT is trained through two losses, both relying on unsupervised tasks. The first loss is the Masked Language Model (MLM) loss, which randomly masks 15% of the tokens in the input sequence using a "MASK" token. The task is to predict the identity of the masked token, using only the neighbouring tokens, from both left and right. In order to avoid a mismatch between training and inference, masking is not only done through the token "MASK" token, which is used 80% of the time, while 10% of the time any sub-word token is randomly changed for a different one and the other 10% of the time it is not modified at all.

The second loss is the Next Sentence Prediction (NSP) loss, and it is aimed at learning a relationship between the two sentences in the input. This can be fundamental for downstream tasks such as Question Answering. To implement this task, when loading the training data, half of the input paired sentences are correctly consecutive ("IsNext") while the other half of the time they are randomly paired ("NotNext"). The NSP loss is a binary loss used to identify if the pair is actually consecutive in the data or not, using the hidden representation of the "CLS" token.

In this thesis we use BERT in most chapters (except in Chapter 7, where we also make use of XLNet (Yang et al., 2019), a similar kind of model) to obtain word- and sentence-level representations of text through the Huggingface library for Python (Wolf, Debut, et al., 2020). In general we use BERT because it has also been preferred in other research in the TTS field, such as, Jia et al. (2021); G. Xu et al. (2020); Kenter et al. (2020).

2.6.2 Deep Spectrum

Originally proposed for sound classification, Deep Spectrum (Amiriparian et al., 2017) is a technique to extract representations for audio, but crucially, by using a spectrogram or mel spectrogram plot. This is a stark contrast from much other recent research proposing speech or general acoustic large pre-trained models to extract representations from waveforms. Deep Spectrum leverages large pre-trained models for image classification. These are very deep models that are task independent. Although in the original paper the authors test different configurations for the feature extraction, in this thesis we use the default options for their released implementation[5], which uses VGG-19 (Simonyan & Zisserman, 2015). VGG-19 is a convolutional image classification model trained with more than a million images.

The procedure to obtain representations starts by extracting either the linear spectrogram or mel spectrogram from the target audio and then plot it with matplotlib library for Python (Hunter, 2007). If only one vector is required for the complete audio sample, then a plot of the entire audio is used. Alternatively, fragments can also be extracted for a more fine-grained representation. Plots are scaled to a fixed-sized square matrix, of 224 x 224 pixels for VGG-19. This image is put through the network in a forward pass. The weight activations of the fully connected linear layer located prior to the output layer are extracted as a feature vector, with a dimension of 4096.

Amiriparian et al. (2017) found Deep Spectrum features superior to knowledge-based ones (such as statistics over amplitude, formants, and other acoustic characteristics) for sound classification. They believe that the main advantage of the Deep Spectrum features is that the image classification models are agnostic, and make no assumptions about the audio or the speech when extracting the representation. Because of this, and the results obtained in previous collaborations (Oplustil-Gallegos et al., 2020; Williams et al., 2020) we use these features to represent acoustic context throughout this thesis.

2.6.3 Global Style Tokens

Y. Wang et al. (2018) proposed Global Style Tokens (GST), an unsupervised method that extracts a style vector from reference speech, jointly-learnt with a TTS system, without the need of an additional loss. The method leverages the reference encoder from Skerry-Ryan et al. (2018), which first extracts a fixed-length vector from a mel spectrogram. This vector is the input to the Style token layer, which finally outputs a style embedding (see Figure 2.12). GSTs were originally proposed for the first Tacotron architecture (Y. Wang et al., 2017), but they can potentially work with any other.

[5] `https://github.com/DeepSpectrum/DeepSpectrum`

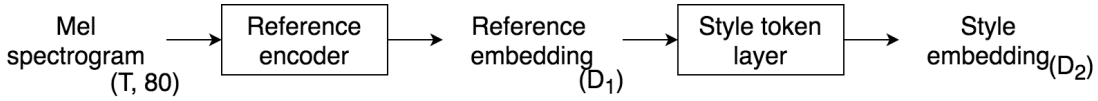

Figure 2.12: Global Style Token architecture. T corresponds to the number of frames and D to a hidden dimension.

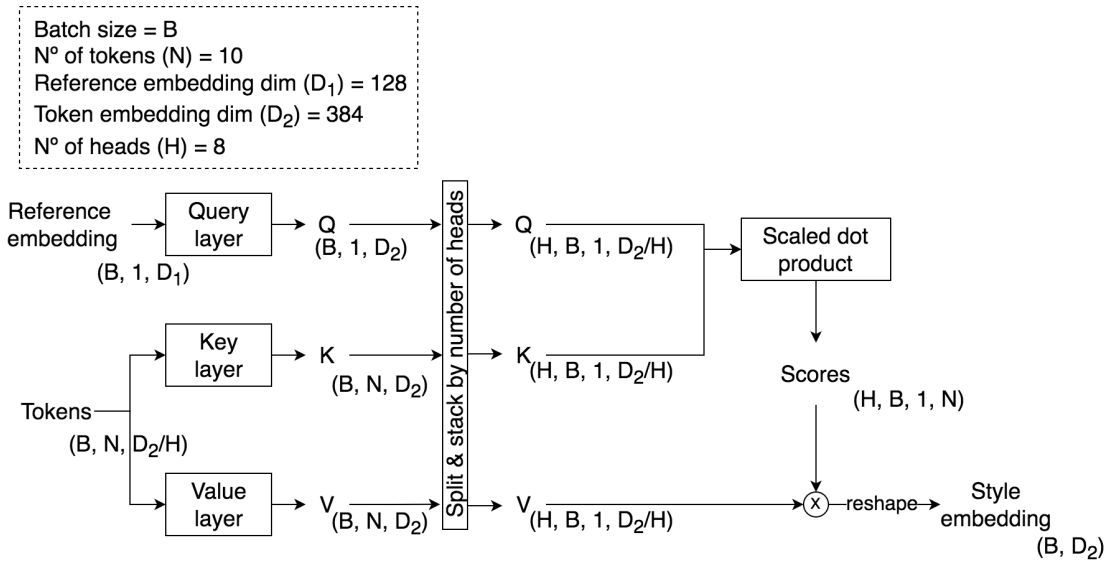

Figure 2.13: Style token layer with example values for illustrative purposes.

The motivation of Y. Wang et al. (2018) to propose GSTs is to replace the need for manual labelling of emotions, styles or expressivity in general, which is costly and error prone. GSTs automatically find soft labels for the speech, by embedding the most salient characteristics. By incorporating this module into an encoder-decoder TTS architecture, the authors hypothesize that the reference encoder should extract anything present in the speech that is not predictable from the text (or from other labels given to the system), such as style characteristics or even noise. GSTs abstract the reference embedding into a set of tokens that act as learnt labels for these characteristics, to finally condition the TTS architecture on it.

The input to the reference encoder is the ground-truth mel spectrogram, e.g. the same as the target for the TTS system. The output is a compressed representation of the mel spectrogram into a fixed-length vector called the reference embedding. The reference encoder is made of six 2D convolutional layers with batch normalization and ReLU activations, and a final unidirectional GRU layer. The last hidden state of the GRU layer corresponds to the reference embedding.

Figure 2.13 shows a detailed diagram of the Style token layer architecture with example settings for illustrative purposes. Our description is based on the implementation by NVIDIA[6] and the original paper. The Style token layer is a multi-head attention module (see Section 2.4.2.5) that finds the relative importance of each token in a set of tokens with respect to the reference embedding, e.g. the tokens are the learnt labels, and we need to determine how relevant each one is for a particular reference embedding.

[6] `https://github.com/NVIDIA/mellotron/blob/master/modules.py`

Each token is a vector of dimension equal to the token embedding size divided by the number of heads (to allow slicing of the matrices later). There are 10 tokens in total, and they are initialized from a normal distribution (with zero mean and standard deviation of 0.5). Crucially, in order for these tokens to behave as soft-labels, they are shared across all training data, i.e. every utterance is compared to the same tokens. After initializing the tokens they are passed through a tanh activation which produces values between -1 and 1.

Following the naming convention for attention matrices, K and V correspond to the set of tokens, while Q is the reference embedding. Each matrix is processed by a linear layer, projecting them all into the same dimension, which must be divisible by the number of heads. The output is sliced and stacked according to the number of heads (8). The scaled dot product of K and Q gives scores that are then multiplied by V. The result is reshaped to obtain a single vector of fixed length for every reference embedding. This style embedding represents the relevance or contribution of each token for that particular reference embedding. The style embedding is concatenated to every element of the encoder output in the original Tacotron paper.

At inference time, there are two main options: a reference mel spectrogram can be input to the GST, and the generated speech will probably simulate the characteristics of the reference (to a certain extent), even if it does not match the input text. This, for example, the authors claim, can perform prosody transfer. Alternatively, the token scores can be manipulated, after empirical verification of what the values are encoding (which will depend on the characteristics of the training data).

For example, by analyzing the tokens one-by-one, the authors found that some tend to specialize in an acoustic characteristic such as fundamental frequency, intensity, or speaking rate, whilst other tokens encode more abstract characteristics such as emotion. When training with added artificial noise, the authors see that different types of noises are learnt as if they were styles, such that a subset of the tokens encodes the types of noise, while another subset encodes the clean speech. When training with multi-speaker data without adding explicit speaker labels for the TTS system, they see that the tokens encode speaker identity.

It can be seen that the method involves several hyperparemeters, including the number of tokens, the number of heads, and the embedding dimensions of the reference embedding and the style embedding. If there is any preliminary idea of how many "soft categories" might be found in the data, this could be used to select the number of tokens. In the original paper, the authors claim that 10 tokens was enough to capture the expressivity of their training data. When experimenting with multi-speaker data, they use 1024 tokens, to try to capture each speaker. In the original paper the authors claim that increasing the number of heads improves the results more than increasing the number of tokens. However, in Kato et al. (2019) where the method is leveraged, it is claimed that too many heads can cause overfitting. The reference embedding dimension determines how much the reference sample should be compressed. The style embedding dimension will most likely depend on the dimension of the location where it will be used to condition the TTS system.

2.7 Some remarks on context in Text-to-Speech

As we claimed in the introduction to this chapter, historically the evolution of TTS shows a trend of incorporating longer spans of within-utterance context in order to seek improvements. While we described the TTS frameworks in previous sections we tried to emphasize this fact. In this section, we briefly recapitulate, so that we can make this trend explicit and use it as a motivation for our proposed method of conditioning TTS systems on local linguistic context.

Context-dependent units were used by both concatenative and HMM-based frameworks for TTS, as described in Section 2.2. For concatenative systems, this implied that data selection aimed for context-dependent diphone unit coverage. The model of within-utterance context in these systems was formalized within the target cost. For HMM-based TTS systems, the trained models are context-dependent phone models. The main issue of context-dependent units/models is that sparsity increases when considering more context. For unit selection methods, this issue was typically handled through back-off methods. For HMM-based models, the solution was to tie models with similar contexts using a regression tree. Furthermore, HMM-based models incorporate the use of smoothing algorithms like MLPG to improve each generated frame by considering the complete output sequence.

The improvements seen in Section 2.4 across the frameworks using deep learning imply the extension of neural network architectures to consider longer spans of context. While at first only one acoustic frame was modelled at a time, the integration of recurrent layers helped to model the relationship between frames, although as explained in Section 2.3.3, recurrent layers need to use special architectures to improve on long dependency modelling. These systems also benefited from the use of smoothing algorithms like MLPG to improve the frame-by-frame continuity.

Sequence-to-sequence models, upgraded with attention, allow encoding of the complete input sequence in architectures such as Tacotron 2, with convolutional filters that model 5 input symbols at the time, and attention mechanisms that (in theory) can find relationships between distant elements in the encoder output and the decoder input. Tacotron 2 auto-regression improves frame-by-frame modelling in the decoder by modelling past within-utterance output. The post-net, the counterpart of the MLGP algorithm, smooths over the output sequence considering all its elements. Transformer-based TTS architectures, such as FastPitch, that use self-attention, allow the modelling of both sequences in its entirety.

We believe that state-of-the-art architectures have now reached the maximum potential of within-utterance context by modelling the complete sequence, and it is time to look beyond the utterance. A possible solution would be simply model more than one utterance at a time. But, as we have discussed throughout this chapter, deep learning architectures often struggle with long sequences, which, together with implementation issues, makes this approach less attractive.

In contrast, our approach consists of conditioning the TTS architecture on a representation of the local linguistic context: the previous utterance. In a way, this is similar to the context-dependent units described above, as now the utterance is context dependent, but through conditioning and the various methods proposed in Chapters 4 and 5, we believe the model can learn the relationship between different utterances and their corresponding context, whilst handling sparsity. This approach, instead of explicitly extending the span of the data, as we would if simply modelling

more than one utterance at a time, aims at extracting the fundamental information from the context that would help the TTS architecture to improve the modelling of the current utterance.

Chapter 3

Methodology

In this brief chapter we give a general description of how we will answer the main research questions stated in the Introduction chapter of this thesis. As described in that chapter, Chapters 4 to 8 contain the experiments, each with its own research questions and/or hypotheses. Importantly, all the experiments and methods in this thesis are designed to be as fully automatic as possible, avoiding manual work that would be costly both in time and money, and would impede the exact reproducibility of the results.

Chapters 4, 5 and 6 (Part I and Part II), where we will develop methods to answer RQ1, will follow a conventional structure, similar to a paper: (1) we state concrete goals, research questions and/or hypotheses; (2) we review related work; (3) we propose some new methods or feature extraction for a baseline TTS architecture; (4) we select a data set or we collect some data if required (in Chapter 6) to train TTS systems; (5) we evaluate them through one or more listening tests to test the hypotheses and/or answer the research questions; (7) we provide further analyses; and (8) we discuss the results and state the contributions to the thesis. In Chapter 4 we prototype a first method to condition TTS on context, which was important to define the main research questions of this thesis and to decide on the next steps. In Chapter 5, we further develop our method with a systematic comparison of different context representations. In Chapter 6 we explore alternative methods and we focus more on data collection, because we use dialogue data.

Chapter 7 and 8 (Part III) are slightly less conventional in their structure as instead of further developing the method, we take an in-depth approach to answer RQ2 and RQ3. In Chapter 7 we make use of multiple data sets with the best method obtained in Part I, evaluating through both listening tests and objective evaluations. Finally, Chapter 8 is dedicated solely to in-context evaluation. As such, we do not train new TTS systems. Rather, systems already proposed in previous chapters are evaluated both through in-context listening tests and a newly proposed in-context objective evaluation.

In the current chapter we briefly review these methodological decisions that are common to more than one of the experimental chapters, related to the baseline architectures (Section 3.1), the data pre-processing and front end analysis (Section 3.2), listening tests (Section 3.3) and the type of analyses we prefer in this thesis (Section 3.4).

3.1 Text-to-Speech architectures

We use two state-of-the-art architectures in this thesis, Tacotron 2 (Shen et al., 2018) and FastPitch (Łańcucki, 2021), which were described in detail in the Background (Section 2.4.2.3 and Section 2.4.2.7, respectively). We selected these because they are representative of the two dominant architectures at the moment: attention-based versus explicit duration prediction. We take as starting point two NVIDIA implementations as we are confident of the accuracy in the architectures and the high quality of the code[1][2]. This is especially important for Tacotron 2 as the original paper does not contain sufficient detail to implement it ourselves. For FastPitch it is very convenient because the published implementation is provided by the original author. Both implementations are open source and based in the PyTorch library (Paszke et al., 2019).

In all our experiments, we train all systems to be compared for a fixed number of epochs or iterations (making sure that by then the model's validation loss has converged), in order to compare them fairly, as this is not a variable we are experimenting with. This is not problematic for FastPitch systems, as we have seen empirically that after convergence the speech generated by later checkpoints changes very little, showing the stability of the model. In contrast, for Tacotron 2 this is a bit more complicated, as empirically we have seen that, even after convergence, the quality of the synthetic speech generated by different checkpoints can be very different (Shechtman, Haws, & Fernandez, 2021). In practice, for industry applications, manual checkpoint selection can be very important in this case. However, as said before, we decided to compare all systems trained for the same amount. Therefore, we do not do manual checkpoint selection.

Mel spectrogram extraction is performed identically for both architectures, as we combine both with the Waveglow neural vocoder (Prenger et al., 2019), described in Section 2.4.3.1, employing a pre-trained checkpoint from NVIDIA[3]. To extract mel spectrograms for all our experiments we use the configuration required by the vocoder to ensure compatibility. All waveforms are resampled at 22050 Hz and normalized. The STFT is applied with filter length = 1024, hop length = 256, and window length = 1024. The mel filters are calculated through the librosa library in Python (McFee et al., 2015), using 80 channels, applied from 0 Hz to 8000 Hz. Finally, the log is taken for dynamic range compression. For FastPitch we also extract fundamental frequency contours using the Praat (Boersma, 2001) library for Python (Jadoul et al., 2018).

3.2 Data, pre-processing, and front end

In this thesis we use several data sets with different characteristics in order to answer our research questions. Table 3.1 summarises all the data sets used in this thesis, with a brief description of their characteristics and the chapters in which they are used. Although we state the overall amount of the available data, for each exper-

[1] `https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/SpeechSynthesis/Tacotron2`

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

[3] `https://github.com/NVIDIA/waveglow/`

Data set	**Style**	**Speakers**	**Hours of data (∼)**	**Used in**	**Source**
LJ Speech	Read	LJ	24	Chapters 4, 5, 7	Ito and Johnson (2017)
Podcasts	Spontaneous dialogs	Speaker 1 Speaker 2 (Podcast 1)	12 (together)	Chapter 6	Private data
Parallel Audiobook	Read	sde ekl mth (Emma book) msm (all books)	7 7 7 17	Chapter 7	Ribeiro (2018)
Spotify podcasts	Spontaneous monologues	Show 1 Show 2	70 40	Chapter 7	Clifton et al. (2020)
IBM Debater	Semi-spontaneous monologues	RG	24	Chapter 7	Mirkin et al. (2018)

Table 3.1: Summary of data sets used in this thesis.

iment we will specify what the train, validation and test sets are. Except for the Podcasts, all data sets are open source. The Podcasts were collected and bought during my internship at Amazon and are core to Chapter 7. All the data sets are for US English, except the Parallel Audiobook corpus, which is UK English.

Notably, because we are investigating the use of the local linguistic context, none of our data sets is comprised of isolated speech utterances. Rather, all of them are comprised of sequences of utterances in the context of larger speech units, such as paragraphs or monologues, from which we can retrieve context-target utterance pairs through the metadata (data set naming conventions) or by storing this information when we segment the data into utterances. LJ Speech, Parallel Audiobook and IBM Debater already come segmented or provide utterance level alignments for segmentation. For the rest of the data sets that are not segmented into utterances, we describe how we did this in the respective chapter where the data set is used.

Because in this thesis we are not focused on front end analysis, we use the most basic and standard procedure in order to obtain a minimally good quality paired transcription for each utterance. LJ Speech, Parallel Audiobook and IBM Debater text comes already normalized. Both Podcasts and Spotify podcasts data sets are automatically transcribed through APIs (Amazon's STT and Google's STT, respectively) for which the output is usually normalized text except for that it may containing numerical symbols. When the output was non-alphabetical the utterances were discarded. To train with Tacotron 2 (Chapter 4) we do not need forced alignment, so we transcribe the data with Festival (Black, Taylor, Caley, & Clark, 1998). To train with FastPitch, data sets need to be forced aligned, and for convenience were transcribed using the dictionaries provided together with the aligner, Montreal Forced Aligner (McAuliffe et al., 2017), using also pretrained

acoustic models for English. Out-of-vocabulary words were transcribed using G2P[4] and added to the dictionary. Punctuation, which MFA strips during alignments, was restored. In both cases, phonetic transcription uses the ARPABET phone set. We do not extract other linguistic features such as prosodic or linguistic features because our focus is on the extraction of representations from the context. The only exception to the procedure described here is for Chapter 6, as being part of an internship, the internal company's systems were used.

3.3 Evaluation implementation

All the main results reported in this thesis are obtained through listening tests, with the notable exception of the results in Section 8.3.4, where we investigate and report the use of an objective in-context evaluation. For some experiments we use objective comparisons such as word error rate for the synthetic speech or validation loss, but only for analysis or to eliminate systems prior to subsequent experiments that will employ listening tests. Listening tests are therefore our main evaluation method in this thesis. For every listening test, we state a set of hypotheses that are later rejected or supported considering the results. We mostly use the listening test designs described in Section 2.5 (together with the corresponding statistical tests) which are state of the art, for evaluating sentences in isolation, and additionally, we research in-context listening tests taking as starting point the work by Clark et al. (2019) in Chapter 8. For each listening test reported in this thesis, we aim to give as much detail as possible regarding its design:

- What kind of test was applied: MOS, MUSHRA-like, preference test. This decision will depend on the hypothesis we are testing and the number or systems compared.
- If we include a reference or not.
- Instructions presented to the participants.
- A clear list of the systems compared.
- How many participants did the test and their relevant characteristics (such as accent).
- How many samples were tested, how they were selected and if the text was presented to the participants or not.
- Methods to discard/filter participant responses to ensure high quality results.

Because of the COVID-19 pandemic all the listening tests in this thesis were conducted online, with the required ethics permission issued by the School of Informatics, number 2020/60160. The tests were implemented on Qualtrics[5] and Prolific[6], was used to recruit and pay participants. Because online results obtained through listening tests might not be as reliable as in person ones, we apply two strategies to

[4] `https://github.com/Kyubyong/g2p`
[5] `https://www.qualtrics.com/`
[6] `https://www.prolific.co/`

ensure high quality results: we filter suspicious results (through methods explained for each listening test) and we pay a fair compensation. For all listening tests we add code to make sure that listeners have listened to all the audio required before moving forward through the test. We always have at least 20 valid participants per test.

The total number of participants per test depended on the budget available and the average completion time that the test required, in order to ensure that all participants received UK minimum wage. As my scholarship does not cover costs for running experiments, the listening tests that will be presented in this thesis were funded by different sources: Chapter 4 listening tests were funded by CSTR; Chapter 5 by collaborator Johannah O'Mahonny's scholarship; Chapter 6 by Amazon; Chapter 7 and 8 by CSTR.

3.4 Data-driven analysis

Throughout this thesis, additionally to seeking improvements from a method to condition TTS systems on context, we aim at providing further understanding of what these methods are doing. Therefore, in Chapters 4, 5 and 6, we try to always provide further insights in the form of analyses after obtaining results, and Chapters 7 and 8 are in nature analytical.

The methods we propose in Chapters 4 and 5 are based on GSTs and attention. In the original paper (Y. Wang et al., 2018), in a *post hoc* analysis the authors noted that when manipulating specific parameters in the tokens, the output would contain modifications of particular acoustic properties, depending on the characteristics of the data set (such as fundamental frequency or speech rate). Manipulating values and inspecting the output is a time consuming process that is not entirely transparent, as not all the values encoded have a clear effect on the resulting speech, making this type of analysis non-trivial. Regarding attention, there is a great debate about how interpretable attention weights really are (Jain & Wallace, 2019; Serrano & Smith, 2019; Wiegreffe & Pinter, 2019). Because attention is inherently encoding the relevance of the items between two vectors, many authors considered that it could be taken as a window into the inner workings of the architecture. One agreement there seems to be in the literature is that attention mechanisms that are applied to vectors encoded through **non-recurrent** architectures are more interpretable. But in general, this is still a research topic in progress.

Considering the above, and my own experience more oriented to linguistic and data analysis, we take a different approach in the analyses in this thesis. Our analyses are predominantly data-driven: first, we train with data sets that have specific characteristics we are interested in (or we manipulate the data to have those characteristics). Then, we make hypotheses of what results we would expect to see given those characteristics and how we believe the methods are working. And finally, given the resulting synthetic speech we confirm or reject the hypotheses and therefore update our understanding about the method. Chapter 7 predominantly has this structure. We believe that this approach provides a more transparent analysis than the ones mentioned above. This type of analysis also prioritizes the final naturalness of the synthetic speech rather than the inner workings of the methods themselves, because although we are interested in proposing new methods, our top priority is to improve synthetic speech naturalness through the use of local linguistic context.

Part I

Main method

Chapter 4

Conditioning Text-to-Speech on context

4.1 Introduction

In this chapter we present the first steps towards conditioning TTS systems on local linguistic context. The goal of this experiment was to test if the proposed method could bring significant improvements to the synthesized speech quality (**RQ1**, from the Introduction). Recall that in the Introduction we limited the span of context to experiment with in this thesis, to the previous utterance with respect to a current utterance, following their original order in the data set. The method we propose conditions the current utterance to the previous utterance both at training and inference time, in a similar way to a bigram model.

In this first method we propose, we begin by representing context through acoustic features only. Mel spectrograms from the previous utterance are processed by a context encoder into an utterance-level representation. Acoustic representations have been widely used to improve TTS (Section 4.2.1), but to our knowledge, we are the first to propose to extract them from context, rather than from the current utterance. By using acoustics we further restrict context span to the past only as, in an application, future context would only be available in written form.

There are different modes to condition the current utterance on the previous one. Crucially, each utterance is originally a sequence with an independent length, as Figure 4.1 illustrates at the top. There is not a one-to-one relationship between the start of two consecutive utterances or the end of both. The only constant relationship is that the context utterance is always preceding the target one. Consequently, simple time-wise concatenation cannot be justified (illustrated in Figure 4.1.A). We consider then that there are two viable options: we can either compress the previous utterance into a single vector through an encoder (as in Figure 4.1.B) or we can maintain the full resolution of the previous utterance and learn the time-wise relationship between the two utterances (as in Figure 4.1.C). In the first case, a unique vector is simply repeated over L and either concatenated or summed to the current utterance. In the latter, we can *learn* a step-wise correspondence between the two sequences, for example, through an attention-scoring mechanism. The scores will guide the concatenation of the previous utterance's step-wise vectors to those of the current utterance.

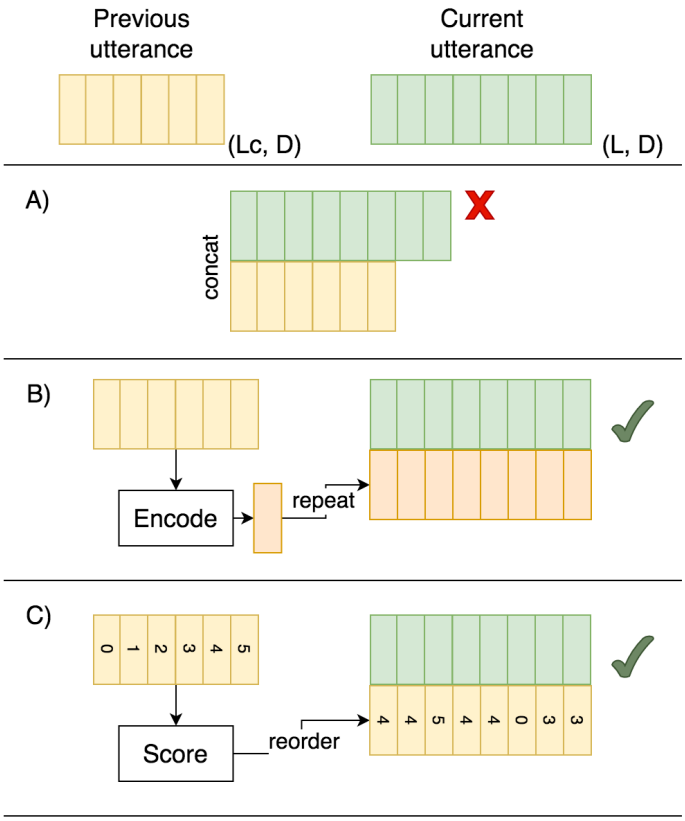

Figure 4.1: Illustration of the problem of conditioning current utterance on the previous one. As shown at the top, each sequence has it's own length, *Lc* for the previous utterance, and *L* for the current one. In this example, L is input symbols (e.g. phones). Part A) shows how time-wise direct concatenation would be performed. Part B) shows how an encoder is used to compress the previous utterance into a single vector that is next repeated and concatenated to every step in the current utterance. Finally, part C) shows the use of a scoring mechanism that would guide the concatenation. Notice how, in theory, the step-wise vectors in the previous utterance could be re-arranged and even repeated if necessary when concatenating to the current utterance. Concatenation can always be replaced with addition.

For the experiments in this chapter we use mode B), as we consider it to be more appropriate for utterance-level acoustic representations. In Chapter 5, both B) and C)'s modes are studied, while systematically investigating the relationship between conditioning methods and context representation.

As stated before, the main goal of the experiments in this chapter is to find significant differences in synthetic speech quality when conditioning on context compared to a baseline that does not use context. By conditioning the TTS system on context we expect that it will learn the relationship between utterances, predicting the acoustic characteristics of the next utterance to synthesize more accurately. However, we do not know if the context encoder will, without additional supervision, provide a representation that encodes the relationship between the two utterances. Therefore, in this chapter we also test whether a multi-task setting is needed in order to meaningfully incorporate context into TTS. We compare the use of context in conjunction with multi-task training to solely the use of context. We also compare two different ways to design the multi-task scenario. In the first case, the additional task consists in predicting an embedding of the current sentence given the previous one. In the second, the additional task is to classify whether or not the utterances are in the correct order.

Lastly, additional to performing a traditional listening test, we consider if our systems require a non-traditional evaluation design and if so how we would conduct it. As stated by Clark et al. (2019), every TTS system should be evaluated in-context, but a system trained with contextual information should perform significantly better in-context than one trained without it. The problem is that in-context listening test designs are very recent and we still lack experience when testing systems in this way. Therefore, the in-context listening test design we performed in this chapter was somewhat exploratory, but it nevertheless allowed us to identify the many challenges of evaluating TTS in-context.

Our results with traditional listening tests show that enhancing TTS with acoustic context is informative for the model, leading to statistically significant improvements in the synthetic speech naturalness. These results, together with the novelty of the concept, encouraged us to pursue this topic as the main research goal of this thesis. The experiments presented in this chapter were uploaded as a pre-print to ArXiv (Oplustil-Gallegos & King, 2020).

4.2 Related work

We identify two areas of related work relevant for this chapter. The first one corresponds to different methods to encode utterance-level acoustic features to enhance TTS during training. The second, regards multi-task learning in order to learn a relationship between two sentences.

4.2.1 Utterance-level unsupervised acoustic encoding

It is known that linguistic features extracted from text alone are not sufficient to predict prosody (Taylor & Black, 1999). Therefore, to learn anything that is not predictable from the text, it is necessary to exploit other sources of information. The same acoustic variability that a TTS system is trained to predict can be directly utilized to improve it. Acoustic encoding can be divided into supervised and

unsupervised methods. Supervised methods are mainly based on manual labelling of data, following a set of categories that gather utterances with common acoustic characteristics. The labels can be used to condition training and inference, expecting that they are informative enough to improve the model's predictions. Although in theory it would be possible to label every acoustic variation in the data, this is usually not feasible. Moreover, it is very difficult to design a set of labels objective enough to ensure perfect annotator agreement (Henter et al., 2018).

Meanwhile, unsupervised approaches do not rely on manual annotations, but rather develop methods to learn to represent acoustic variation in an automated way. After reviewing the literature, we identify two main categories for this approach: (1) the use of reference encoders in conjunction with latent spaces, and (2) automatic clustering. We will limit the description to methods applied at the utterance-level only, as this is the scope of the experiments in this chapter.

4.2.1.1 Reference encoders and latent spaces

A reference encoder takes as input a mel spectrogram and outputs a fixed-dimensional reference embedding, regardless of the length of the utterance. In Skerry-Ryan et al. (2018), one of the first reference encoders for TTS was proposed. The encoder is designed with a bottleneck architecture aimed at capturing the prosodic characteristics of the input. The hidden state from the last layer is extracted as the single vector to represent the utterance. In Skerry-Ryan et al. (2018), this vector is directly concatenated to every step of the TTS text encoder outputs, conditioning the decoder. Because at inference time the mel spectrogram of the utterance to synthesize is not available (this is exactly what we are trying to predict), an embedding is obtained by providing a suitable mel spectrogram from a *reference utterance* to drive the encoder. Because during training time the mel spectrogram processed by the encoder corresponds to the target of the MSE loss, this encoder learns an embedding that conditions the model to synthesize speech with the similar acoustic characteristics as the mel spectrogram input to the reference encoder.

The use of a reference encoder can be enhanced by explicitly learning a latent space for the obtained reference embeddings. GSTs, described in detail in Section 2.6.3, do this by learning weights for a set of randomly-initialized embeddings with respect to a the fixed-length reference embedding extracted by the reference encoder. A similar approach is the use of a Variational Autoencoder (VAE) to learn a latent space for the reference embedding. A VAE has an encoder and a decoder. The encoder generates a z vector for the input, for which a distribution is learnt. This is the latent space. During training, the decoder consumes a z vector randomly sampled from the latent space from which it must learn to reconstruct the original input through a reconstruction loss (for more details on VAEs, see Section 6.4).

Y.-J. Zhang et al. (2019) use a VAE to learn a latent space for the output of the same reference encoder proposed by Skerry-Ryan et al. (2018). The embedding from the reference encoder is passed through two parallel fully-connected layers which generate the mean and standard deviation used to derive the latent vector. This vector is then summed to the text encoder outputs, conditioning the decoder. As before, a suitable reference utterance needs to be selected at inference time, or alternatively, manual control can be achieved by sampling directly from the latent space through heuristics. Crucially, both of the use of GSTs or VAEs methods provide a further level of abstraction of the reference encoder output. Because this

outperforms direct conditioning using the reference embedding alone (Y. Wang et al., 2018) (i.e. without going through the explicit latent space), we believe that this can be taken as evidence of the sparsity of acoustic representations obtained by reference encoders.

We can see how inference is non-trivial for these methods. Even if manual control is desirable for many applications, for many others it is necessary to drive synthesis automatically (for example, in the case of real-time dialog systems). Moreover, Watts et al. (2015) showed that random sampling of reference utterances is not preferred by listeners. While for our experiments we make use of an acoustic encoder in conjunction with the GSTs, in contrast to the methods described above, we do *not* use the same mel spectrogram as input to the encoder and as training target. Our method uses the previous utterance's mel spectrograms as input for an acoustic encoder, and therefore, instead of denominating it a reference encoder, it is a context encoder. Meanwhile the mel spectrogram for the current utterance remains the training target. At inference time, therefore, we provide a self-supervised solution to the inference problem. We do not need to look for a reference utterance to drive the encoder, because that utterance will simply be the context utterance.

At the time of running these experiments, the only work that proposed something similar was Tyagi et al. (2020). They propose a TTS with a VAE that encodes reference utterance acoustics. They explore the use of different methods at synthesis time to sample from the VAE latent space when synthesizing multi-utterance texts. They measure the acoustic distance of the previously synthesized utterance with respect to the candidate reference utterances in a 2D space. This distance is considered in a general formulation to find the best reference utterance. However, this technique is applied at inference time only, while we propose the use of context for both training and synthesis.

Finally, although Y.-J. Zhang et al. (2019) claims that the use of VAE outperforms GSTs in a subjective listening test (especially for non-parallel style transfer), in this chapter we preferred GSTs, among other reasons, because this method does not require additional losses. This fits well with our experiments, as we will be able to design *ad hoc* losses to encourage GSTs into learning the relationship between context and current sentence.

4.2.1.2 Automatic clustering

Utterance-level manual labels can be replaced by automatic clustering of acoustic features. Classical unsupervised clustering, such as K-means or hierarchical clustering (Eyben et al., 2012), can be applied to pre-process the data before training. The clusters found are taken as labels, which are utilized in the same way as if they had been manually obtained to condition the TTS at training and inference time. In Eyben et al. (2012), multiple frameworks to cluster data for TTS are tested. Listeners were asked to judge if the clustering over the natural data was correct, achieving more than a 70% accuracy.

Classical clustering techniques happen offline, independently from TTS training. In contrast, latent spaces, like the ones described in the previous section, can be designed as forms of soft-clustering, learnt jointly with the TTS. In the case of the GST framework, because the tokens are shared across all training data and are randomly initialized, they can be regarded as unsupervised categories. Conditioning through the style embedding thus corresponds to the weighted sum of the

unsupervised categories with respect to every sample in the data.

The VAE framework has been notably extended by Van den Oord et al. (2017) to VQ-VAE. In contrast with VAE, VQ-VAE decodes a discrete representation of every sample obtained by a quantized latent space over the encoder outputs. In a nutshell, an encoder outputs a set of vectors z for one sample. The latent space is expressed as a set of embeddings K, each one associated with an index. The distance between each vector z and each embedding K is obtained, next collecting the indexes of the nearest embeddings. The gathered indexes are fed into the decoder. To learn the embeddings, rather than using the gradients of the reconstruction loss, two new terms are introduced. Both terms aim at minimizing the error between the K embeddings and the encoder outputs, and vice versa. It can be seen how the embedding vectors are behaving as clustering in the model, especially if a reduced number is used, and/or if the encoder outputs are low dimensional. Similarly to GSTs, the set of K embeddings is shared across all the training data. Although most of the papers we found where VQ-VAE is used, model sub-utterance units (for example, finding phone identities), the same method could be applied to utterance-level representations. Unsupervised clustering can be seen as complementary to the methods described in the previous section. We believe that a context encoder can benefit from the implicit use of clustering. We do not know how many types of context-target relationships exist or how sparse they are, and therefore including clustering, or soft-clustering, as part of the method can mitigate this risk. This is a further reason to utilize GSTs for our experiments. Even if VQ-VAEs might outperform GSTs, they require two additional terms for the model's loss. As stated before, because we want to design our own losses, GSTs still seem the best option as their original formulation does not require supervision.

4.2.2 Learning the relationship between two sentences

A fundamental aspect of our method relies on learning relationship between two utterances. Although we could not find examples of such work within the TTS field or for other speech related tasks, this has been a topic of research for the fields of neural machine translation (NMT), natural language generation (NLG) and neural language modelling, where they aim to learn a relationship between two sentences (text only).

For example, BERT (Devlin et al., 2018), as described in Section 2.6.1, is not only trained with the most widely known Masked Language Modelling loss, but also optimizes for a Next Sentence Prediction (NSP) loss. The inputs to BERT correspond to paired sentences, joined by a "separation" token, together with a vector that indicates which tokens belong to which sentence. During training, 50% of the time the two sentences are truly consecutive as in the training data, while the other 50% of the time the pairs are selected randomly. The NSP task consists of a binary classification, deciding whether the sample is a true pair or not. Devlin et al. (2018) explicitly state that the aim of this task is to learn the relationship between two sentences, and they show that it is especially beneficial for downstream tasks such as question answering.

For NLG, Jernite et al. (2017), design losses that can capture supra-sentential relationships to train sentence encoders. The losses rely on self-supervised tasks which leverage the structure of the data, and that are aimed at capturing discourse

relationships between consecutive sentences. They were the first to propose the NSP task used by BERT, although they call it an "order task". It was originally design to capture relationships between two sentences where the second one is an elaboration of the first one. They also propose the "next task", where a sequence of three consecutive sentences is collected, together with a different set of five candidates from later in a paragraph. The task consists of deciding which of the candidates is the fourth sentence in the sequence. They observe that the best results are obtained when training with all the tasks jointly, improving metrics on downstream sentence classification.

In a different approach, relying on the use of attention to interpret context, Tiedemann (2017) experiment with extended context for NMT. In one setup, they use as input the source sentence to be translated, preceded by the previous source sentence, where each token is marked as context. They see that the model is actually capable of learning that the translation should correspond to the second sentence only, and that the contextual tokens are attended 7.1% of the time only. They find it hard to extract systematic patterns on how the model is leveraging the context, but they see that it improves resolution of ambiguous pronouns.

Our experiments including the multi-task setting explore the use of losses to learn the relationship between the context-target pairs for TTS, inspired by some of the losses proposed by Jernite et al. (2017). In Chapter 5 we also make use of attention to find relevant context relationships from text features at the word level, similar to Tiedemann (2017).

4.3 Method

4.3.1 Context-target utterance pairs

Our method is designed to work with data sets that are naturally structured as longer speech-text units. Only in this way, the natural data might contain supra-sentential relationships that we can exploit for TTS. In contrast, data elicited in the form of independent, randomized sentences, as the Arctic script (Kominek & Black, 2004), is not suitable for our method. We do not consider this a short-coming of the method, as the field is moving towards the use of more natural and spontaneous training data, in order to achieve higher naturalness and expressivity. For example, audiobooks are a popular source of training data for TTS (Zen et al., 2019). Indeed, audiobooks are an excellent example of the kind of data appropriate for our method: they are originally structured into chapters and paragraphs, which tend to be recorded following the true order of the text (e.g., there is no randomization of the sentences before recording). Even if it is necessary to split the data into memory-manageable utterances for TTS training, the original structure is usually kept as part of the meta-data. For example, as is the case of the LJ Speech data set (Ito & Johnson, 2017), every utterance is named by enumerating it, including the chapter it belongs to. While most TTS training simply ignores this information during training, we propose to leverage it. We can consider the work by Peiró Lilja and Farrús (2018) as a close attempt to make use of the inherent structure of the data, by incorporating paragraph-level prosodic features, given a previously analyzed data set. However, in our work we do not focus on where the utterances are in the major structure, but rather the dependency between two consecutive utterances.

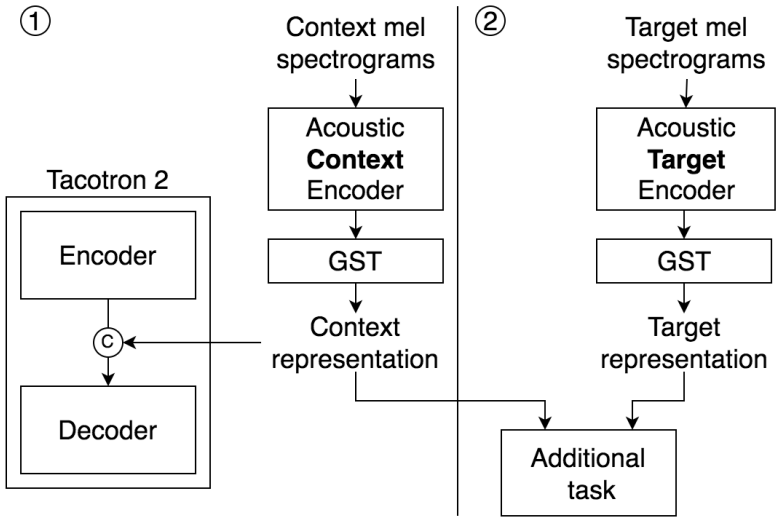

Figure 4.2: Proposed method. The method is divided into two components. In the first one, on the left, the extraction of a context representation is used to condition a Tacotron 2 base architecture. The second one, on the right (optional) an additional task is performed to encourage the context representation to learn the relationship between context and target, by performing an additional task using representations of both.

We leverage the meta-data to collect context-target utterance pairs within each chapter using regular expressions over filenames to ensure the pairs are truly consecutive. This is handled during the data loading process during training. For example, a context+target pair would be "LJ001-0017"+"LJ001-0018". As these are prototype experiments, we simply discard any utterance that does not have a valid previous utterance (such as the first utterance of every chapter).

4.3.2 Using acoustic context

The implementation of our method consists of two distinct components. The first one extracts a representation from the context, which is used to condition the TTS model. An acoustic encoder and GSTs are employed to obtain the representation. The second one considers the multi-task setting, where we test two different designs. The base TTS architecture corresponds to Tacotron 2 (described in detail in the Background, Section 2.4.2.3). The complete method is illustrated in Figure 4.2. We divide the method into these two components to clearly show that, optionally, it is possible to utilize only the first part, the conditioning, without the multi-task setting, or both together.

4.3.2.1 Conditioning on previous utterance acoustics

The first component of the method consists of conditioning the prediction of the current utterance on a representation of the previous one. Once the context-target utterance pairs have been identified, we load the mel spectrograms for both sentences. The mel spectrograms of the context utterance are used as input for an Acoustic Context Encoder (ACE) (see Figure 4.2, left side). Our ACE has the same architecture as the reference encoder in Skerry-Ryan et al. (2018), but crucially, as

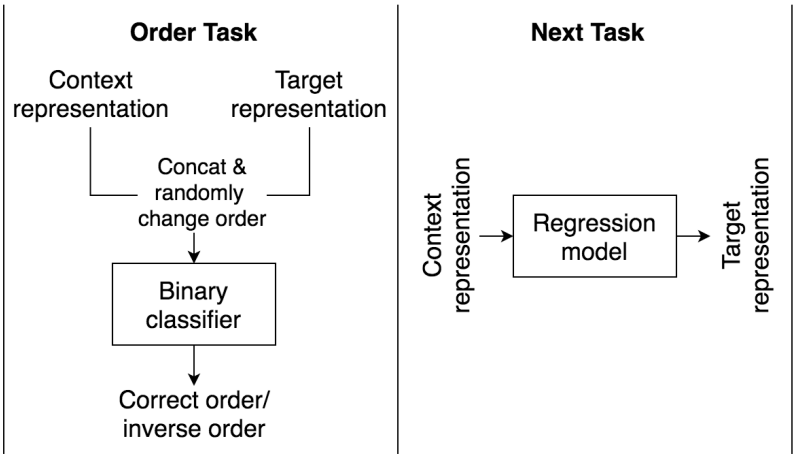

Figure 4.3: Additional tasks. On the left, the binary classification Order Task. On the right, the regression Next Task.

stated before, the input mel spectrograms belong to the context utterance. This means that the target mel spectrogram does not match the one processed by the ACE. We believe that this mismatch should encourage the model to learn the relationship between the context-target pair, even if there is no additional supervision.

The output of the ACE is a single vector that is the input to a GST module. We use 10 tokens with 8 heads and an embedding size of 256, as suggested in (Y. Wang et al., 2018). The context representation obtained is then repeated to match the length of the Tacotron 2 encoder outputs, and concatenated to it. The decoder input dimension is increased to match this dimension. We condition Tacotron 2 in this location, as it has shown to successfully drive the model, e.g. (Skerry-Ryan et al., 2018; Y. Wang et al., 2018). At synthesis time, in a real-world application e.g. the context would correspond to the previously synthesized utterance; in the experiments in this chapter we use ground-truth context (teacher forcing). We made this decision in order to prototype the method under ideal conditions. In later chapters we analyse the impact of using synthesized context (Chapters 7 and 8).

4.3.2.2 Additional task

We compare two different formulations of an additional task (see Figure 4.2, right side). Both require the model to find a relationship between context and target utterances. Because we want to drive the learning of the context representation (the GST output), the relationship has to be established at that point. Therefore, we also obtain a representation for the target utterance, in the same way as described in the previous section, using an Acoustic Target Encoder. Notice that this representation is never used directly for TTS, but it is only needed as part of the additional task.

As Figure 4.2 shows, we have both a context representation and a target representation, both in the same space, and obtained from the acoustics, but through two independent instances (no weights are shared) of an acoustic encoder. These two representations are the input to the additional task. In this work we tried two tasks, illustrated in Figure 4.3.

Order task: the two representations are concatenated, with a probability of 0.5, in the right order (keeping the previous-current sequence), or otherwise, in the wrong order. The binary classifier determines if the order has been preserved or not. We expect to capture an order dependency between the two sentences in this way. The task uses a binary cross entropy loss that is then summed to the Tacotron 2 main loss. The classifier is comprised of 3 layers, halving the dimension of the input at each one ([512, 256, 128, 64]) up to a final output layer with dimension 1. We include ReLU activations and batch normalization after every layer.

Next task: the context representation is used to predict the target representation in a regression task. An additional MSE loss is summed to the main Tacotron 2 loss. This should directly encourage the context representation to encode information that can predict the acoustics of the target sentence. The regression model has a bottleneck architecture with 5 layers, with dimensions [256, 128, 64, 128, 256], followed by ReLU activations and batch normalization.

4.3.3 Data and tools

As stated in Section 4.3.1, as training data, we require audiobooks that are relatively complete, in order to collect as many context-target sentence pairs as possible. This is why we prefer the LJ Speech data set over other options such as LibriTTS (Zen et al., 2019), which has been highly filtered. Moreover, we prefer to test the method within a single speaker at the moment. Before starting the experiments, we confirmed that the enumeration provided in the meta-data of the utterances truly matched the order of the content in the original books, by inspecting a set of utterances chosen randomly for a few of the available chapters. LJ Speech consists of a total of 7 books, which we split into 12604/311/290, train/validation/test sets respectively.

As mentioned in the Methodology (Section 3.1), our base architecture is a Tacotron 2 implementation by NVIDIA[1] and a GST implementation also developed by NVIDIA[2]. All models were trained from flat start until reaching 100k iterations (when validation loss had converged). As a vocoder, we use an available Waveglow model pretrained with LJ Speech[3].

4.4 Evaluation and results

Given our concerns, stated in the introduction to this chapter, regarding the appropriate evaluation method, we decide to perform a two stage evaluation. First, we run a traditional MUSHRA-like design with isolated sentences, and later a forced choice in-context design.

The first test helped to short-list the models to be tested in-context. Because in-context evaluation inherently requires participants to listen to twice as much audio (context duration plus target duration), we preferred to have a minimum number of systems for this test, to avoid listener fatigue. Moreover, we believe that our proposed method should improve synthetic speech both in isolation and

[1] `https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/SpeechSynthesis/Tacotron2`

[2] `https://github.com/NVIDIA/mellotron/blob/master/modules.py`

[3] `https://catalog.ngc.nvidia.com/orgs/nvidia/models/waveglow_ljs_256channels`

in-context, therefore we expected to find significant differences over the baseline for both evaluations.

As described in the Methodology, all tests were implemented in the Qualtrics platform and participants were recruited through the crowed-sourcing platform Prolific. Participants were screened as US residents and native speakers of English, and paid through the platform. As mentioned earlier, in Section 4.3.2.1, at synthesis time, all systems that use context utilize ground-truth acoustics of the previous sentence.

4.4.1 Isolated sentences naturalness test

We seek to answer three questions through this evaluation: **RQ4.1**, does the use of context through the proposed method significantly improve naturalness over systems that do not use context?; **RQ4.2**, for the systems that use context, can an additional task bring significant improvements?; and **RQ4.3**, for the systems that use context and an additional task, which of the two tasks is better? We compare five systems in a MUSHRA-like test comparing isolated sentences:

- Vocoded natural speech
- Baseline
- ACE only
- ACE + Order Task
- ACE + Next Task

Natural speech is vocoded to serve as the hidden reference. Our baseline is a Tacotron 2 system without any use of context. We test our proposed method with 3 different systems: ACE only, which corresponds to the first component of the method only, conditioning without an additional task. ACE + Order Task and ACE + Next Task test the full method with the two components, comparing the two proposed additional tasks.

We design a MUSHRA-like task were 20 participants were asked to score from 0 to 100 the naturalness of 20 sets of samples and to find the vocoded hidden reference by giving it a score of 100.

Results are shown in Figure 4.4. We test for normality with Shapiro-Wilk, and therefore, test for significance with Wilcoxon signed-rank with Bonferroni correction (α=0.05). **All pair comparisons are significantly different**, with the ACE + Next Task model scoring the best.

Therefore, with respect to the questions stated at the start of this section, we can say that: (**RQ4.1**) because all the systems using context were significantly better than the baseline, the use of context through the proposed method does improve synthetic speech naturalness. Then, regarding (**RQ4.2**) and (**RQ4.3**), the incorporation of an additional task to supervise the context representation can improve the results, depending on the task formulation. This, because although ACE + Next Task was significantly better than ACE only, the ACE + Order Task was not, indicating a problem with its design, which can even diminish the model's performance.

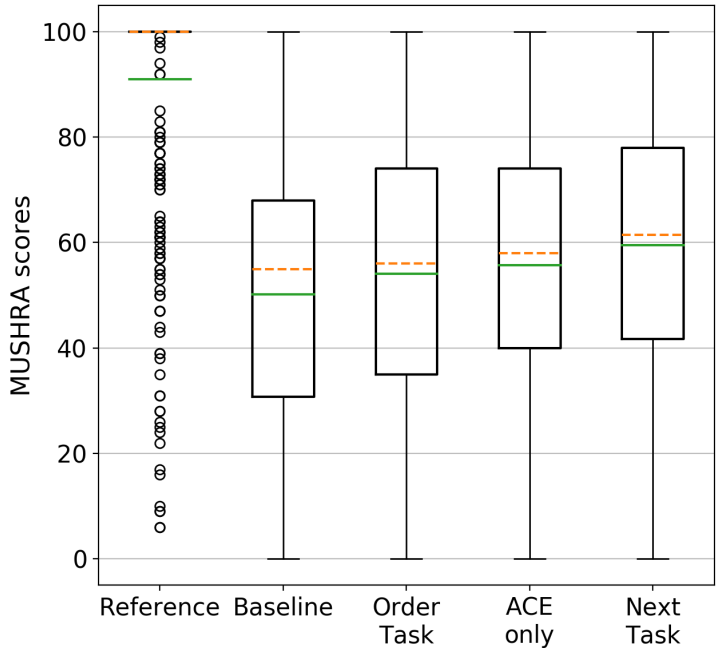

Figure 4.4: Isolated sentence naturalness MUSHRA-like test results. Boxes extend over the inter-quartile range of the scores. Solid lines correspond to the mean; dashed lines to the median. Circles represent outliers.

Moreover, this means that ACE only, without an additional task, can, to some extent, exploit the context by simple conditioning. This is very important as it can be taken as evidence that there is information that is salient enough between context and target, such that the GST encoded it without supervision. We hypothesize that if the ACE + Order Task did not bring improvements, it might be that the context and target acoustic representations extracted are too close in the embedded space. This might make the classification task too difficult.

4.4.2 In-context preference test

Having found the best system through the first evaluation stage ACE + Next Task, we want to know: **RQ4.4**, is the system truly learning a relationship between the context and the target sentence? We hypothesize that, if this is the case, then the system should be preferred in an in-context evaluation.

To provide contrast points to the ACE + Next Task system, and understand better the results, we include other three conditions in the preference test. These are the conditions, and our hypotheses with respect to the ACE + Next Task system:

ACE only: the second best system in the naturalness test. Because this system does not include a loss designed to capture the context-target relationship, we hypothesize that it should perform worse than the ACE + Next Task system.

Random ACE: we train a new system for this test. This one is trained using the same two component method just as ACE + Next Task, but instead of providing the correct previous-current pair of utterances (truly consecutive in the training data) at training time, we provide pairs of non-sequential utterances only, by randomly

selecting a context utterance from any other chapter than the one of the current utterance. We hypothesize that the ACE + Next Task should be preferred over Random ACE, as the first one is trained with the ground-truth previous-current utterance pairs which should be more informative than randomly selecting pairs.

Reference: finally, we include ground-truth vocoded context-target. We hypothesize that the Reference should be preferred over the ACE + Next Task system, as it corresponds to natural ground-truth speech, where the speaker has not only acoustic context in mind but also the matching semantic context to perform the most appropriate prosody.

The test is designed as a pairwise forced choice preference test, where each of the three conditions described above is compared against the ACE + Next Task system. Participants are presented with two pairs of context-target utterances at the time, from two different systems. Figure 4.5 shows an example of each screen presented to the participants.

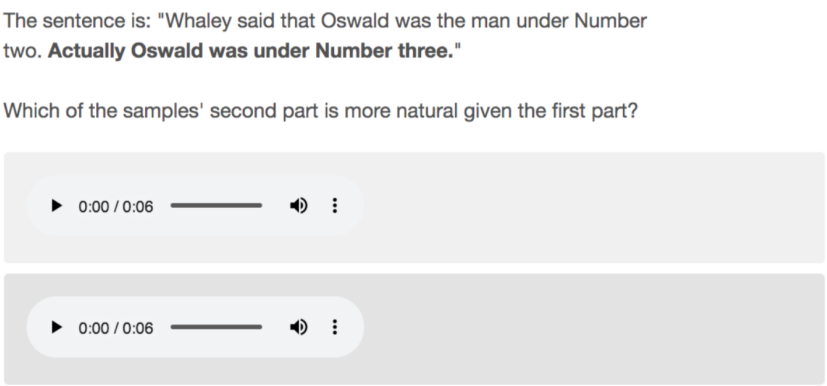

Figure 4.5: In-context pairwise forced choice evaluation screen capture. On the top, the text matching the audio is presented, where the target sentence is on bold. The participant needs to select their preferred sample.

For all the conditions, the ground-truth context is used both for synthesis and is vocoded to be listened as the context by the participant. Context and target utterances are concatenated using a fixed silence of 0.5 seconds to join them. Sample selection was not straightforward: considering we were presenting two utterances at the time, we tried to select pairs of samples that had a complete meaning. We limited the length of the combined samples to a maximum of 10 seconds, to avoid listener fatigue. We used 10 pairs of utterances where the target finished in a full stop. We recruited 30 participants, that at the start of the test received the exact instructions below:

- You will be presented with two audios at a time: both of the same sentence.
- The first part of both audios is identical, however, you will notice that the second part will be slightly different.
- We want you select, for each pair, which second part is the most natural sequel to the first.

Results are shown in Figure 4.6. The binomial test was applied to compare the three conditions to the ACE + Next Task system. As expected, ACE only

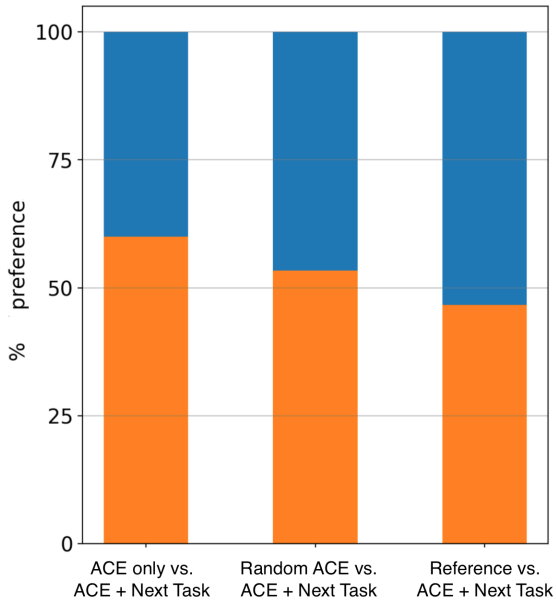

Figure 4.6: In-context pairwise forced choice evaluation results. ACE + Next Task system corresponds always to the bottom bar.

was significantly worse. However, the other two conditions were not significantly different from the ACE + Next Task. In the next section, we make an in-depth analysis to shed some light on these unexpected results.

4.5 Analysis

There was no significant difference in preference between the Random ACE condition and the ACE + Next Task model for the in-context evaluation. We expected that the Random ACE condition synthesized speech should be affected by the inconsistency in context-target pairs that were not truly sequential. What is the Random model learning from context? Moreover, is the Random ACE model using context at all?

In this section, we use a qualitative method to analyse the use of context by the models. We run synthesis of the same target utterance multiple times, but each time with a different context (randomly selected). If a model truly leverages context, the speech generated at every run should be different. If the model is ignoring the context, synthesis will always be identical.

Before proceeding with the analysis, we need to make a relevant digression. At inference time, there are two sources of stochastic behaviour in our implementation: first, Tacotron 2's pre-net uses drop-out at inference time. Because the pre-net is run for every frame during the auto-regression, the results can vary considerably when synthesizing the same test multiple times. Second, Waveglow uses random sampling to generate waveforms, and can produce different results for the same mel spectrogram. Therefore, to correctly proceed with our analysis, we need to control for these two factors. To factor out the vocoder's effect, all the samples in this section are synthesized with the Griffin-Lim algorithm.

Is the effect of drop-out at inference time actually significant? We compare fundamental frequency contours to illustrate the effect. See Figure 4.7 for the plots.

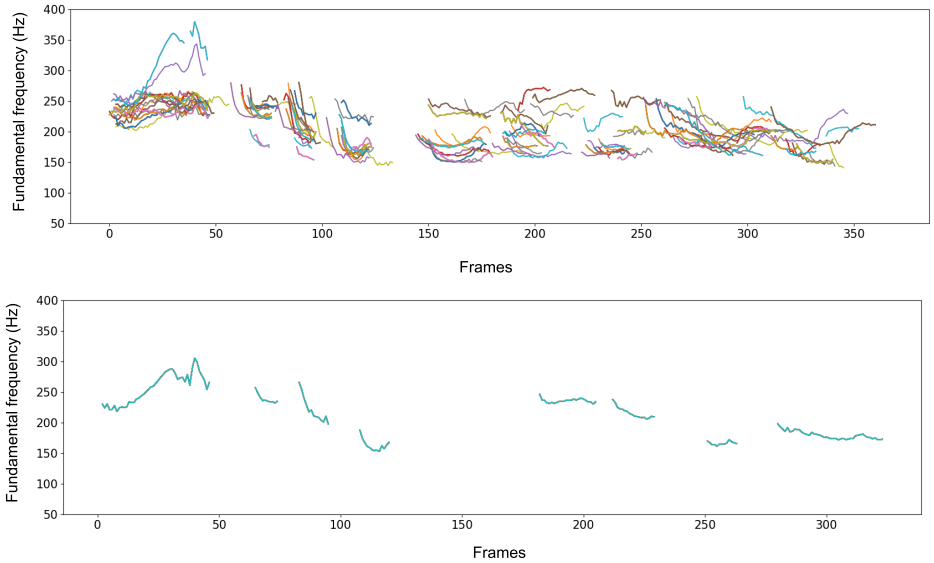

Figure 4.7: Illustration of the effect of drop-out at synthesis time. The baseline TTS system is run 50 times to synthesize the same utterance. At the top, *with* drop-out, and at the bottom, *without*.

The top plot shows the same utterance synthesized 50 times, with different random seeds using the Baseline system, *with* drop-out. The bottom plot shows the same 50 runs *without* drop-out. By turning drop-out off, the system synthesizes always the same output. Moreover, we can see that drop-out enriches the pitch variance for the utterance, yielding very different prosodic patterns, including different duration. Therefore, to remove any variation due to drop-out, we will turn it off during the analysis in this section, to clearly see the effect of context.

We can now proceed with the analysis of the effect of context, without the interference of drop-out or the vocoder. We compare ACE + Next Task with Random ACE (See Figure 4.8). The plots show two relevant characteristics: first, the Random ACE (bottom) results in the same output every time, even when changing the context, implying that context is ignored. Second, for the ACE + Next Task model, the use of context provides information that mostly shifts the pitch pattern to higher or lower values, or stretches it in time. Only some of the runs show differences on emphasis location, for example, towards the end of the sentence (some sentences end with a falling pattern while others with a flat one). Consequently, the question is, if the Random ACE is not making use of context at synthesis time, why was ACE + Next Task not significantly better than it, in the in-context evaluation?

Remember that the samples in Figure 4.8 are not the ones evaluated in the listening test, as those were obtained using at inference time drop-out and the neural vocoder. In contrast, if we compare the same systems but now using drop-out, we obtain the plots in Figure 4.9. We can see that both models present great variation, with greater duration differences for the ACE + Next Task. Both models seem to vary more than the Baseline, when compared to Figure 4.7.

Given this evidence, we conjecture three possible reasons (complementary) for Random ACE versus the ACE + Next Task results in-context. First, our method

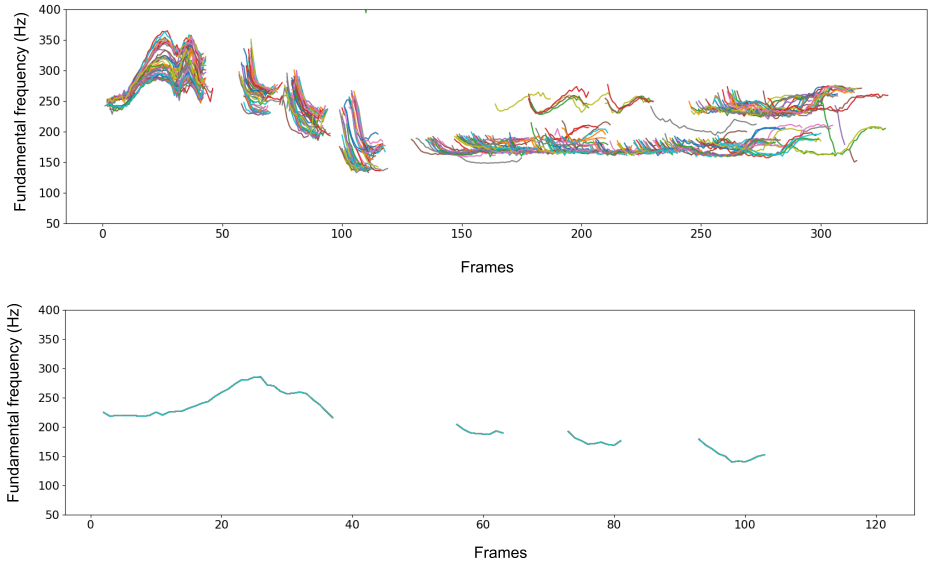

Figure 4.8: Comparison of the ACE + Next Task (top) and Random ACE (bottom) use of context, both *without* drop-out. Both systems are run 50 times to synthesize the same sentence, each time using a different context.

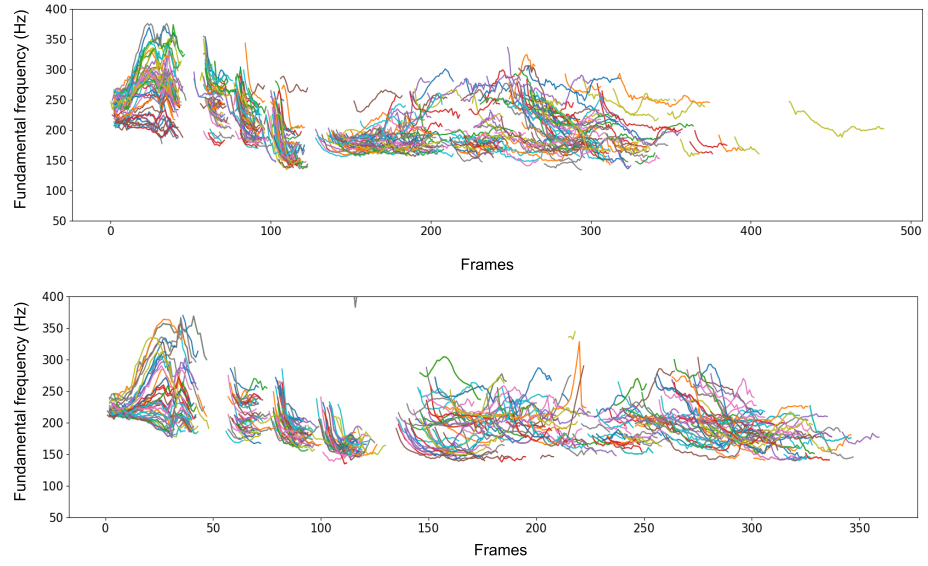

Figure 4.9: Comparison of the ACE + Next Task (top) and Random ACE (bottom) use of context, both *with* drop-out. Both systems are run 50 times to synthesize the same sentence, each time using a different context.

needs to be refined to optimally make use of context to predict the most appropriate prosody. For example, we can consider the use of textual features too. Second, it is possible that the use of context for the Random ACE, at some stage of training, drove the model to a different solution than the Baseline, even if ending up ignoring the context. The increased prosodic variation obtained could have been preferred by the listeners. Lastly, from a different perspective, maybe the result points to a sub-optimal in-context evaluation design. This is supported by the fact that the Reference was not significantly better than the ACE + Next Task model.

Finally, to be thorough, we inspect using the same analysis, how the ACE only model uses context. Figure 4.10 illustrates the system for both *without* (top) and *with* (bottom) using drop-out. It can be seen on the top plot, that the use of context results strictly in a global shifting of the pitch pattern, with minimal duration differences. This contrasts with the effects of context for the ACE + Next Task system, in Figure 4.8. The strictly global effect of context is probably kept when using drop-out, showing less overall variation than the ACE + Next Task system. We hypothesize that these differences made ACE + Next Task significantly preferred in both tests over ACE only.

We reach two important conclusions from this analysis. Even if we try to analyse the differences between the systems, it is evident that inference time drop-out interacts with the effect of context, and therefore, it is better to avoid this kind of system for future analysis. Lastly, a system that leverages thoroughly the use of context will result in both global and local changes in the predicted fundamental frequency, as well as affecting duration.

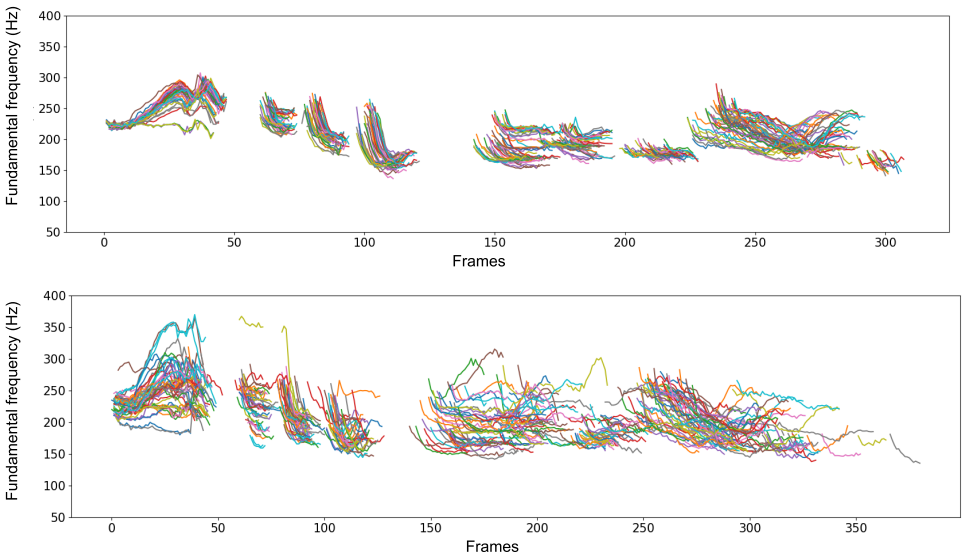

Figure 4.10: Comparison of the ACE only system *without* (top) and *with* drop-out (bottom). The ACE only system is run 50 times to synthesize the same sentence, each time using a different context.

4.6 Conclusion and next steps

The main conclusion from the experiments in this chapter is that the use of context can improve synthetic speech quality. Leveraging the structure of the data to find local contextual dependencies, in the form of previous-current (or context-target) sentence pairs, is possible. Direct conditioning on a context representation improves over a baseline, while the use of a correctly designed additional task to supervise the representation further improves results. This prototype experiment served to identify all the relevant variables that we will need to take into account throughout this thesis. In the next paragraphs, we briefly describe them. Crucially, these identified aspects helped us delineate the research questions stated in the Introduction chapter.

TTS Architecture: although Tacotron 2 is a reliable state-of-the-art architecture, it presents the issues described in Section 4.5, which can complicate the analysis of our method. In order to avoid these issues, further experiments will make use of architectures that are deterministic at synthesis time and that are not auto-regressive. Moreover, considering the issues that have been identified in the literature for attention-based systems (Sections 2.4.2.3 and 2.4.2.6), we will explore architectures with explicit duration prediction.

Context representation and conditioning method: while in this chapter we represent the context through acoustic features only, we will systematically compare the use of textual features. Other work published at the same time as when the experiments here were performed (Guo, Zhang, Soong, He, & Xie, 2021; P. Xu et al., 2019), shows the benefits of including textual features derived from context. Moreover, there are more dimensions to explore regarding context representation: what resolution should the representation have (e.g. utterance-level, word-level, other), or, can we use large pre-trained models to process context to help with possible sparsity? We experiment with these in Chapter 5. Moreover, in the introduction of this chapter we described different options to condition the TTS system on the context representation. We believe that conditioning methods and context representations should match. The two methods proposed in Figure 4.1 are, therefore, tested in Chapter 5.

Multi-task training: although our best system in this chapter included the use of an additional task, in the next chapter we prefer to use ACE only. Despite it being a simpler method, it nevertheless brings significant improvements. Moreover, we prefer to avoid the use of additional tasks because they require more hyperparameters that need to be correctly fine-tuned to obtain optimal results, and because in the following chapter we make use of FastPitch (Łańcucki, 2021), which already comprises three losses.

The "first utterance problem" and using synthesized acoustic context: when collecting the context-target utterance pairs we need to handle the "first utterance", for which there is no context. While in this chapter we simply dismiss it, for future implementations we will handle it. We think that this can be managed by including a vector equivalent to a "start" representation, similar to the start tokens widely used for language modelling in NLP. We describe this in detail in Section

5.4.2. As mentioned before, in this chapter we used ground-truth speech as context. In a real-world scenario, such as an audiobook reader or a dialog system, the context for our method would be synthesized speech (e.g., the utterance previously synthesized). If we synthesize a long piece of text, such as the chapter of a book, it can be seen as a utterance-level regression. We need to consider if there is any possible degradation of the results by using synthesized speech, considering that it will be a mismatch to the teacher-forced training. We will investigate this and discuss in detail in Chapters 7 and 8.

Data domain and speaker: in this chapter we used the LJ Speech data set. The most important reason to choose this data, was to have complete audiobook chapters (minimum amount of missing utterances, to obtain the maximum amount of context-target pairs). In the future, however, we will explore other data sets that are more spontaneous. We believe that our method should bring larger improvements for realistic data, where the speakers might be building stronger context dependencies. We consider the use of more realistic data from Chapter 6 onward. Furthermore, it is possible that some speakers might be more proficient than others in building context dependencies through out their speech.

Chunking criteria: because our method is based on the concept of "previous" and "current" utterances, we need to take in to account how utterances are split for TTS training, and how this might impact the effectiveness of the method. Training data is split (or chunked) into utterances that are of manageable length for TTS architecture and memory constraints. This is more straightforward for data sets that have a ground-truth written form, such as audiobooks, where punctuation is often used as the chunking criterion. If we want to explore other data sets such as conversation or spontaneous speech we will need to think about splitting criteria and how this might affect the context representations. We explore this in Chapter 7.

In-context evaluation: finally, as we mentioned in the analysis section of this chapter, we think the in-context evaluation design can be greatly improved. Although a topic complex as this one could be the sole goal of a thesis, we will dedicate Chapter 8 to researching further how we can contribute to the development of in-context evaluation.

4.7 Chapter contribution

The work presented here has already been used by other authors. Cong et al. (2021) tested a unified system to improve conversational TTS for Chinese. Part of the presented approach consisted of our two part method (with the Next Task). Because they apply it to conversational data, they enhanced the Acoustic Context Encoder with a Gradient Reversal Layer connected to a speaker classifier, that prevents the context representations from encoding speaker identity. The system that used our proposed contextual method with their extension was significantly better in their subjective evaluation. For more details on this paper, see our review in Section 9. We highlight this paper here because it is relevant for us to see that other authors can implement our proposed method and also find that it brings significant improvements to their TTS systems.

Finally, we want to reiterate the many-fold contributions of this initial experimental chapter. We obtained preliminary and encouraging results towards our first research question in the Introduction chapter, seeing that we can obtained improvements by conditioning TTS systems on local linguistic context. To our knowledge, we were the first to propose the use of acoustic representations extracted from local linguistic context to condition TTS systems. Moreover, we identified many relevant variables to experiment with, which were distilled into the research questions proposed in our Introduction.

Chapter 5

Representing context

5.1 Introduction and related work

The experiments presented in the previous chapter provide evidence to suggest that conditioning TTS on **a** representation of local linguistic context can improve synthetic speech naturalness. Now, we recognise that there are multiple possibilities for how to represent context, and that it is possible that some of these representations can be better than others, or that they can encode different types of information. Moreover, those experiments focused on a representation of context derived from acoustic features only -mel spectrograms- from the previous utterance. We did not consider any information derived from the previous utterance's text.

Therefore, the main goal in this chapter is to systematically compare different ways of representing context. Considering that the contrast between acoustic/textual features is a clear starting point, and that in the previous chapter we already reviewed acoustic encoding, we will start by reviewing possible approaches to exploit textual features from context utterances. The experiments and results we present in this chapter were published in Oplustil-Gallegos et al. (2021). Section 5.4 includes novel content.

Several authors have proposed to extract information from text beyond the current utterance. The main difference lies in how these features are obtained. Many authors rely on, either through supervised or semi-supervised approaches, features extracted from text with models or parsers trained with manually-labelled data. Some of these approaches include: condition on position of utterance inside a larger unit such as a paragraph (Farrús et al., 2016; Peiró Lilja & Farrús, 2018), explicit features derived from discourse-level parsing such as discourse relations (Aubin, Cervone, Watts, & King, 2019) or topic structure (Hirschberg, 1990). Other approaches include directly labelling emphasis (Malisz, Berthelsen, Beskow, & Gustafson, 2017; Suni et al., 2020; Shechtman, Fernandez, & Haws, 2021), dialogue acts (Sridhar, Syrdal, Conkie, & Bangalore, 2011), or phrase breaks (Rendel et al., 2017; Klimkov et al., 2017). Most of these approaches extract features that are used to augment the TTS input. While direct labelling can be useful for controllability, accurately predicting labels only from text is hard. In contrast, work parallel to ours has proposed representing neighbouring utterances using large pre-trained language models, extracting *learnt* embeddings to condition TTS systems on. These are the works we focus on in the next section.

5.1.1 Unsupervised text-derived contextual representations

In order to overcome the issues with supervised approaches, unsupervised representations of the text of neighbouring utterances has been proposed, based on the use of large pre-trained language models such as BERT (Devlin et al., 2018). Although these embeddings are not as transparent as the supervised features above, they have been shown to be informative for TTS (Kenter et al., 2020). Moreover, this approach is closer to the acoustic encoding approaches we explored in the previous chapter. These methods had been proposed concurrent to our work in this thesis.

Guo et al. (2021) conditions a Tacotron 2-like baseline on a Conversation Context Encoder for Chinese conversational TTS. This encoder takes as input a set of utterance level embeddings from BERT for 10 sentences into the past (a dialogue history), including the current sentence. The embeddings are extracted for each utterance individually. The output of the Conversation Context Encoder is a fixed-length unique vector which is summed to the Tacotron 2 encoder outputs. Their results show that the use of the Conversational Context Encoder improves over their baseline. They do not specify if BERT is fine-tuned to extract the embeddings.

G. Xu et al. (2020) proposed using BERT embeddings from neighbouring sentences -two past and two future with respect to the current sentence- to improve the prosody of synthetic speech for audiobook data. These embeddings are called cross-utterance vectors, and are utilized to condition a Tacotron 2 architecture. They test two different methods to obtain BERT embeddings (without fine-tuning) for a set of utterances. In the first of which, two embeddings are obtained: one for past and current sentences, and one for the future ones. The two vectors are simply concatenated, and the resulting vector is further concatenated to every encoder step of the TTS system. For the second method, an embedding is obtained for every pair of sentences. In order to concatenate these to the TTS systems' encoder outputs, they use a multi-head attention layer. This layer takes the paired sentence embeddings as keys and values. The outputs of the phoneme encoder for the current utterance are the queries. The multi-head attention outputs are concatenated to the corresponding phoneme encoder step outputs. Although the TTS systems are trained with contextual information from past and future sentences, they evaluate their models with a MUSHRA listening test, which presents utterances one at the time, i.e. in isolation, as we did in Chapter 4. Their results show that the second method had the best results, with better durations, pitch and stress for the synthesized speech.

We consider these approaches are relevant related work for the experiments in this chapter. However, the main difference is that both papers explore the use of contextual sentences beyond the previous utterance only, e.g. our approach is **local**. They both use a different span of context, with Guo et al. (2021) encoding 10 sentences to the past, and G. Xu et al. (2020) a window of 2 around the current utterance, without further experimentation on why these context spans are appropriate or not. The method we present in this chapter to leverage textual context was inspired by G. Xu et al. (2020), but adapted to our scope of local context only. For a further comparison of the latest works in the field making use of textual context representations see Section 9.2.1.

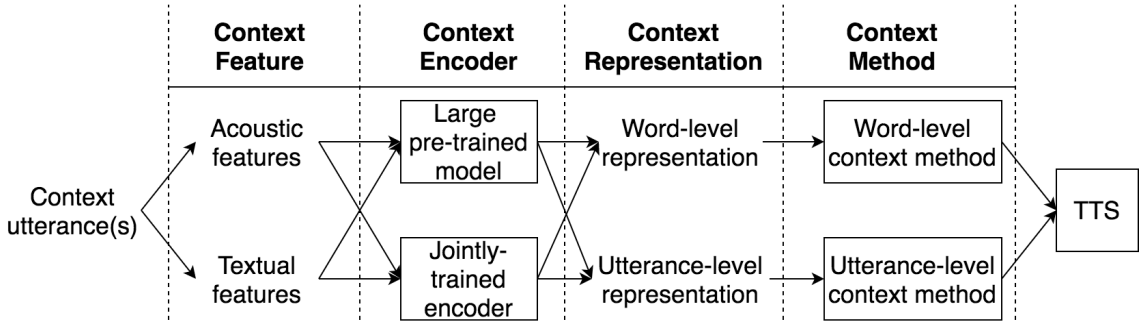

Figure 5.1: Four-stage procedure that captures every approach to represent context.

5.1.2 Research questions

Unsupervised methods to leverage textual information beyond the utterance are closer than supervised ones to our approach for acoustic information presented in Chapter 4. Therefore, we will only consider these for the rest of the chapter. If we compare the method we proposed in Chapter 4 to the papers by G. Xu et al. (2020) and Guo et al. (2021), we can identify some differences of interest (besides the evident acoustic/textual contrast) as well as a common structure for method design. We propose a four-stage procedure that captures every approach to represent context, which is illustrated in Figure 5.1 from a context utterance (or utterances, either past or future ones, depending on the approach), a Context Feature is extracted, which is then fed to a Context Encoder. The output from this is a Context Representation that is then given to a Context Method that finally injects the contextual information into the TTS. Each individual approach, such as G. Xu et al. (2020) and Guo et al. (2021), can be understood as making a design choice for every one of these stages. We can also use this procedure to compare different approaches.

Context Features correspond to the primary information extracted from the context. Our previous work and the related work has made use of either acoustic or textual features. Possibly other types of features could be extracted (perhaps, emotions or speaker embeddings) but we will focus on these two only. A *Context Encoder* is used to extract a Context Representation from the Context Features. While G. Xu et al. (2020) and Guo et al. (2021) use a large pre-trained model such as BERT, having as input textual features, our work in Chapter 4 uses a jointly-trained encoder (the Acoustic Context Encoder), having as input acoustic features. However, in theory, these could be combined the other way around: we could use acoustic features as input for a large pre-trained model for acoustic information, and learn a Context Representation from text through a jointly-trained encoder. *Context Representations* will have a specific resolution: an utterance-level representation will correspond to one vector representing the complete context utterance, while a word-level representation will include one vector per word in the context utterance. There could be other possible resolutions: frame-level, syllable-level, etc. but for simplicity, we will consider only these two resolutions here. Both our work in Chapter 4 and Guo et al. (2021) make use of utterance-level representations only. A context encoder can be adjusted to produce both a word-level or an utterance-level representation. In order to finally inject the contextual information to the TTS architecture, there is a *Context Method*. The context method will specify, for example, if concatenation or summation of the Context Representation is used, in

which location of the model it happens, and will usually include further layers of processing of the Context Representation. For example, our Context Method includes the use of GSTs as a further processing of the Context Representation. The Conversation Context Encoder from Guo et al. (2021) would be considered in this procedure as a Context Method, as well as the cross-utterance encoder from G. Xu et al. (2020). Even the most simple Context Method will use at the very least a linear layer to project the Context Representation to the required dimensionality. Usually, these methods will depend on the Context Representation resolution, matching it to produce a an output suitable to inject into the TTS model.

Considering this procedure to analyse approaches to obtain context representations, we propose a set of experiments to compare systematically the design choices we identify regarding how to represent context. Recall that the main goal of this chapter is to find if there are context representations that, when used to condition TTS systems, can improve synthetic speech naturalness significantly more than others. Considering the main goal and the differences of interest we have identified for comparing the approaches to obtain context representations, our research questions are:

- **RQ5.1**: Would context representations extracted from acoustic features, or context representations extracted from textual features bring most improvement to synthetic speech?
- **RQ5.2**: Would context representations obtained with large pre-trained models, or context representations obtained with jointly-trained encoders bring most improvement to synthetic speech?
- **RQ5.3**: Would context representations at the utterance, or at the word level bring most improvement to synthetic speech?

5.2 Experiments

Our experiments were designed considering the different possible combinations available in Figure 5.1. Although not all combinations are possible (for example, as explained before, Context Methods need to agree with Context Representation resolutions), we try to consider as many as we can, including methods that have not been used so far in the literature, such as word-level context representations. In this section, we explain in detail each of the 8 conditions derived from the proposed procedure. But first, we describe our baseline architecture and how it interacts with the Context Method.

Given the issues we identified in Section 4.5 regarding the transparency of the functioning of Tacotron 2, the experiments in this chapter use FastPitch (Łańcucki, 2021) as baseline TTS architecture. FastPitch is a fast and stable architecture, for both training and inference, and has an open source implementation from the original author[1]. For a detailed description of the FastPitch architecture, see Section 2.4.2.7. To condition FastPitch on previous utterance context, the Context Method is located as shown in Figure 5.2. The Context Method output is summed to the

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

encoder inputs, which in turn corresponds to the sum of the embedded input symbols and positional embeddings. We tested different locations to inject the Context Method output, including summation of it to the input of the Duration Predictor or the Pitch Predictor only, but the first one produced the best results. We believe this is due to the Encoder Transformer's learning power making use of the context representation together with the other inputs. We found that summation gave better results than concatenation in preliminary experiments.

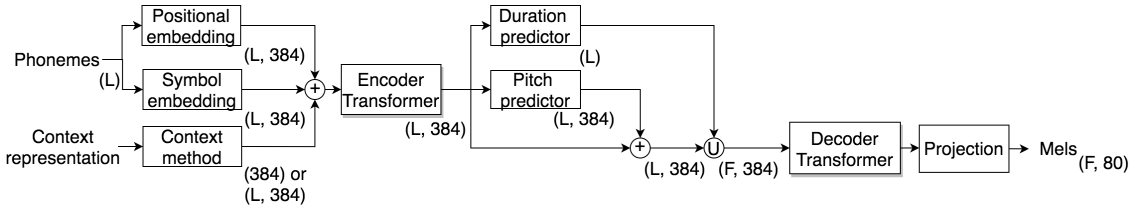

Figure 5.2: FastPitch with Context Method. L stands for number of phones and F, number of frames. U corresponds to upsampling operation.

5.2.1 Context Features, Encoders and Representations

We obtained eight different conditions to test by combining relevant Context Features, Context Encoders and Context Representations. To simplify the description, we divide them into two groups: four conditions based on acoustic features and four conditions based on textual features. Recall that all these procedures are performed for the context utterance. The context utterance corresponds always to only the previous utterance with respect to the current utterance during training or synthesis time, as given by the data set structure.

5.2.1.1 Acoustic feature-based conditions

Condition 1
Mel spectrograms + jointly-trained encoder + utterance-level
Mel spectrograms are obtained from the context utterance, using the exact same process as for the TTS training data (see Section 3.1). The jointly-trained encoder corresponds to the same ACE architecture in Chapter 4 (with no additional task). The output is a fixed-length vector that encodes globally the context utterance, therefore with an utterance-level resolution, just as in Chapter 4.

Condition 2
Mel spectrograms + jointly-trained convolutional block + word-level
Mel spectrograms are extracted through the same procedure as in Condition 1. The jointly-trained encoder corresponds to a convolutional block. This block has the same architecture as the feed-forward block at the end of the Transformer in FastPitch (see Section 2.4.2.7), 1D conv → Relu → 1D conv → summed to the residual → layer norm. To obtain word-level representations, we average the convolutional block output for every word according to word durations extracted during data processing.

Condition 3
Mel spectrogram plots + Deep Spectrum + utterance-level
In this condition, in order to test the use of a large pre-trained model that encodes acoustics, we choose Deep Spectrum. This model, explained in detail in Section 2.6.2, takes as input mel spectrogram plots. These pictures are scaled to a fixed square size and run through a large image classification model, from which the activations at a specific layer are extracted and utilized as a Context Representation. We use the implementation from the original authors[2], using layer *fc2* of the VGG-19 model to obtain, in this case, a 4096-dim vector that encodes the complete context utterance. We choose this model as representative of large pre-trained models, as we found in previous experiments that these embeddings can represent global acoustic characteristics (Oplustil-Gallegos et al., 2020).

Condition 4
Mel spectrogram plots + Deep Spectrum + one second windows
In order to obtain a Context Representation equivalent to the one in Condition 3, but at the word-level, the procedure we followed is the same, except that instead of using a plot of the complete utterance, we use one second windows. Each of these windows is then processed with Deep Spectrum as explained before. Although we know that these do not correspond exactly with a word-level alignment, it is nevertheless a resolution more fine grained than the utterance-level, which is the contrast we are interested in.

5.2.1.2 Textual feature-based conditions

Condition 5
Phonetic transcription + jointly-trained encoder + utterance-level
Phonetic transcriptions are obtained as for all the training data, using 47 symbols comprising phones and punctuation. Word or syllable boundaries are not included in the transcription. A jointly-trained encoder with an initial embedding table layer but with the exact same architecture as for Condition 1, is used to extract a fixed-length vector that represents the whole context utterance.

Condition 6
Phonetic transcription + jointly-trained convolutional block + word-level
To obtain an equivalent word-level context representation from phonetic transcriptions, we use the same convolutional block as for Condition 2. To obtain word-level context representations, the output is averaged over all phones in each word.

Condition 7
Tokens + BERT + utterance-level
As large pre-trained encoder we use BERT. BERT pre-processing tokens (Section 2.6.1) are used as textual derived features. BERT embeddings are extracted using an off-the-shelf model from the Transformers Python library (Wolf, Chaumond, et al., 2020). We do not fine-tune it. A 768-dim vector at the utterance-level is obtained for every context utterance by averaging the activations of the second to last hidden layer.

[2] `https://github.com/DeepSpectrum/DeepSpectrum`

Condition 8
Tokens + BERT + token-level
For the final condition, we obtain word-level context representations through BERT by summing the activations of the last four layers of the model for each token. The same token pre-processing is used to obtain the inputs. As in the previous condition, we do not fine-tune BERT but rather use it off-the-shelf.

5.2.1.3 Discussion

Although some of these decisions might seem arbitrary, we designed the conditions in order to make the different components easily interchangeable to implement and fair to compare. For example, we wanted to use the same jointly-trained encoder architecture for both acoustic and textual derived features at the utterance level. In the same way, an encoder that outputs a fixed-length vector is designed to obtain a global representation, and therefore to obtain meaningful representations with a resolution more fine-grained, we use a convolutional block.

Our past successful experience with Deep Spectrum made us select it for the large pre-trained encoder condition based on acoustic features. In preliminary experiments we tested other models such as VQ-wav2vec (Baevski et al., 2019) (which would have been an appropriate counterpart to BERT, both being based on the same technology), but this was not better. We also tested the use of fundamental frequency contours and SLAM labels (Cosi, 1993) as Context Features, but none performed better than mel spectrograms.

For textual features, in one case the features are phonetic transcriptions and for other textual tokens. We decided to use a phonetic transcription for the jointly-trained condition rather than textual words or tokens as it seemed unlikely that the Context Method would be able to learn a relationship over sparse combinations of words for our training data. We extract BERT embeddings for individual sentences. Although we could have done it for pairs of sentences (leveraging the NSP loss), in this way the comparison to the acoustic-based representation is fairer. This is also why we did not focus on replicating Guo et al. (2021) or G. Xu et al. (2020) both who include the BERT embedding of the current utterance. This is not something we can include in order to be comparable to the conditions based on acoustic features, for which there would not be information at inference time. Finally, to refer easily to these conditions in the next sections, Table 5.1 provides a summary and a short name for each one.

5.2.2 Context Methods

As explained before, depending on the resolution of the Context Representation, word or utterance-level, we combine it with a Context Method that matches it. This means that for word-level representations, the Context Method will output a processed representation at the word-level that is finally upsampled to the phoneme-level to allow summation to the other encoder inputs. In contrast, if the Context Representation is at the utterance-level, the Context Method final output will be one vector only. For both cases, the output will match the hidden dimension of the model. Recall that the main goal of the Context Method is to make it possible to inject the contextual information into the main TTS architecture.

Cond	Short name	Context Feature	Context Encoder	Context Representation
1	mel-utt	Mel spectrogram	Jointly-trained encoder	Utterance
2	mel-word	Mel spectrogram	Convolutional block	Word
3	DS-utt	Mel spectrogram plots	Deep Spectrum	Utterance
4	DS-word	Mel spectrogram plots	Deep Spectrum	One second window
5	Phone-utt	Phonetic transcriptions	Jointly-trained encoder	Utterance
6	Phone-word	Phonetic transcriptions	Convolutional block	Word
7	BERT-utt	Text tokens	BERT	Utterance
8	BERT-word	Text tokens	BERT	Tokens

Table 5.1: Summary of the conditions.

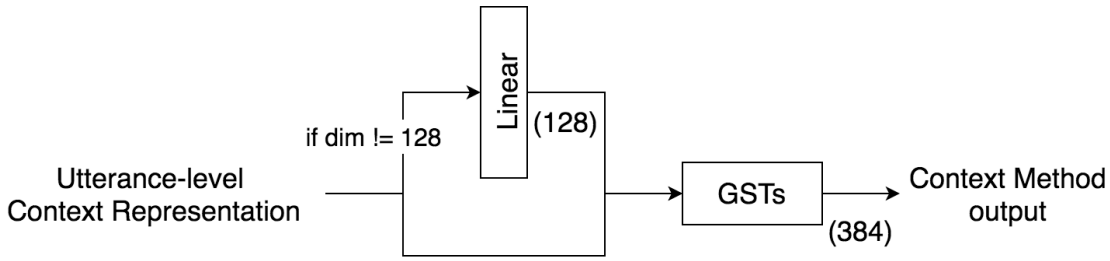

Figure 5.3: Utterance-level Context Method.

5.2.2.1 Utterance-level Context Method

The utterance-level Context Method uses Global Style Tokens which, as we have already shown, can be used to represent context at the utterance level (see the experiments in Chapter 4). GST, in this case, takes as input the Context Representation. In order to use the GST equally for all four different conditions at the utterance level (Conditions 1, 3, 4 and 7), the Context Representation extracted with a large pre-trained model passed through a linear layer, reducing the dimension to 128 (recall that BERT embeddings are 768-dim and Deep Spectrum, 4096-dim). The Context Representations obtained with the jointly-trained encoders, the output is already set to 128-dim.

We train GST with 10 tokens and 8 heads to output a 384-dim vector (the hidden dimension of FastPitch). The output of GST is the output of the utterance-level Context Method, which is finally added to the encoder inputs of FastPitch, as shown in Figure 5.3. We use the GST implementation provided by NVIDIA[3].

[3] `https://github.com/NVIDIA/mellotron/blob/master/modules.py`

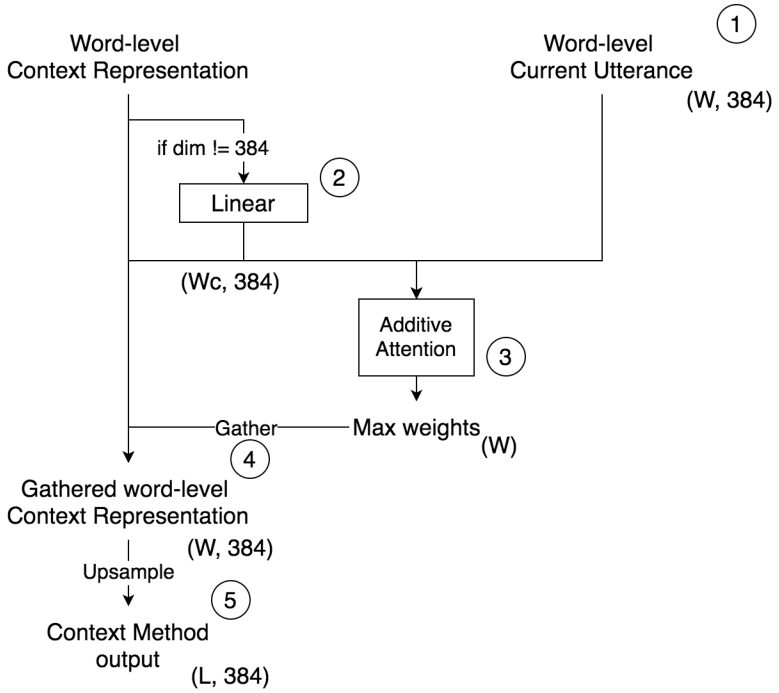

Figure 5.4: Word-level Context Method.

5.2.2.2 Word-level Context Method

The goal of the word-level context method is to find words in the Context Representation that are relevant to the current utterance. The problem we face is the one described in Chapter 4, Section 4.1, where there are an arbitrary number of words in the context utterance, and a different number in the target utterance. None of the related work or our previous work has made use of word-level Context Representations, and therefore we had to develop a new approach. Inspired by one of the methods in G. Xu et al. (2020), we use attention to find the hidden relationships, in this case, word-to-word (rather than phone-to-utterance, as in their work). For a further discussion of the method described in this section, see Section 5.4.4.

Once the word-level Context Representation has been obtained (Conditions 2, 4, 6, 8), the following procedure is applied (follow Figure 5.4):

Step 1: to obtain a matching word-level representation for the current utterance, we average the encoder inputs using the phone-level word lengths of the current sentence.

Step 2: the word-level Context Representation is reduced to the hidden dimension of the model (384-dim) with a linear layer, for the context representations extracted with large pre-trained models.

Step 3: both current and context representations are passed to an additive attention module. The module calculates the relationship between each word-level representation in the current utterance and each word-level representation of the context. Recall that, although these representations come from different spaces (the current utterance is encoder inputs averaged; the context utterance from one of the conditions), the attention uses linear layers to encode the keys and queries into the same space and measure the scores. The output of the attention is a set of the indices of the word-level context representations that had the highest score for each

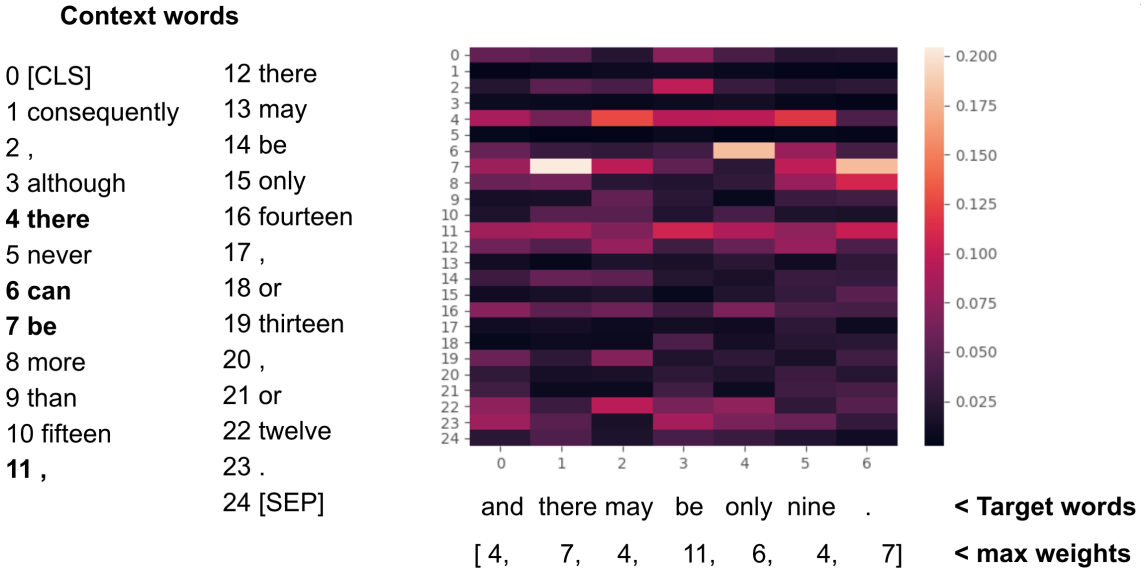

Figure 5.5: Word-level Context Method attention example. Target words include punctuation as a separate item.

corresponding word-level representation from the current utterance in that position. See Figure 5.5 for an example of the condition **BERT-word**. In this example, the current utterance (target words) has 7 items while the context representation has 24 tokens. The attention output was [4, 7, 4, 11, 6, 4, 7], which means that the word with index 4 of the context utterance was the most relevant for the 0th word in the current utterance, and so on.

Step 4: the indexes are used to gather into a new vector the word-level representations of the context utterance that correspond to the words in the current one. Following the same example, the new vector will gather first the word-vector with index 4 and will continue through following the indices. Notice that it might gather repeated vectors (such as the 4th) and that the order in which these word vectors were in the context representation is irrelevant.

Step 5: finally, this new vector is upsampled to the phone length in order to allow summation with the encoder inputs of FastPitch.

5.2.3 Data and tools

We trained and tested all models using LJ Speech (Ito & Johnson, 2017), with 12443 training sentences and 525 test sentences, training from flat start for ∼77k iterations. Following the description in the Methodology chapter, all data was phonetized while force-aligning with the Montreal Forced Aligner (MFA) (McAuliffe et al., 2017) to extract the ground-truth durations required to train the duration predictor, and to obtain the word boundary information needed for word-level representations. We obtained fundamental frecuency contours using Praat for Python (Jadoul et al., 2018), as in the original FastPitch. Just as in Chapter 4, we follow the data naming structure to obtain context-target sentence pairs. To vocode the generated mel spectrograms to waveforms, we used the WaveGlow (Prenger et al., 2019) checkpoint included with the FastPitch implementation, which has been trained on the LJ Speech corpus.

5.3 Evaluation and results

We train one system per condition summarised in Table 5.1. In this section we evaluate the naturalness of the systems through listening tests. Our hypotheses are:

- **H1**: given the evidence in (Guo et al., 2021; G. Xu et al., 2020), together with our experiments in Chapter 4, either text or acoustic features alone should improve the synthetic speech naturalness, possibly bringing different types of improvements.
- **H2**: combining both acoustic and text features will bring further improvements.
- **H3**: considering that the large pre-trained models had been trained with massive amounts of data, they will provide better representations than encoders jointly-trained with the TTS model.
- **H4**: word-level context representations will be better suited for textual features, while utterance-level context representations will be best for acoustic features.

In order to test **H2**, additionally to the models for the already-described 8 conditions, we will train models combining textual and acoustic features. However, to avoid training all combinations of textual and acoustic conditions, we will first compare each separately, and identify the best per group. Therefore, we conducted a three part listening test:

1. Comparing the acoustic feature systems.
2. Comparing the text feature systems.
3. Comparing the best systems in (1), the best systems in (2), and two systems that combine both.

All three listening tests used a MUSHRA-like design and compared 4 conditions, plus the baseline (vanilla FastPitch) and the hidden reference (vocoded natural speech). Each MUSHRA screen presented on the top the reference audio and below, the 6 samples to be rated, without showing the transcription.

Participants were instructed to rate the naturalness of the synthetic speech. The same 25 sentences were used for all listening tests. For the systems using contextual representations extracted from acoustics, features were extracted from the ground-truth context utterance: Section 5.4.2 comments on the possible effects of using synthetic speech instead.

We implemented the test online using Qualtrics and recruited participants who self-identified as native speakers of English and US citizens, using Prolific. A new group of participants was recruited for each listening test. Results from participants who rated any hidden reference lower than 50, or were too fast to complete the task (e.g. the completion time was less than the total duration of all the audio), were discarded. For each test, the first 20 participants who passed these checks were used to calculate the results.

Statistical significance was determined using the Wilcoxon signed-rank test with Bonferroni correction. Figure 5.6 shows the results for the three tests.

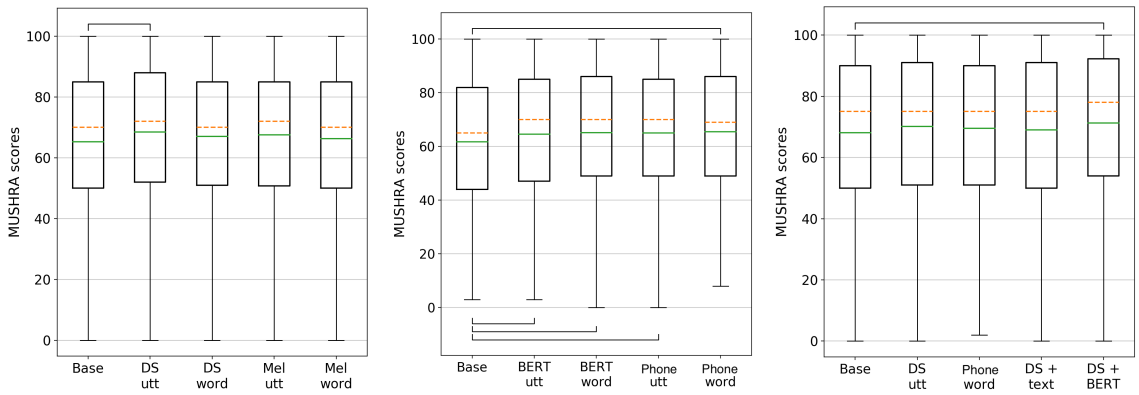

Figure 5.6: MUSHRA-like listening test results, from left to right: (a) comparing conditions based on acoustic features; (b) comparing conditions based on text features; (c) best models compared with acoustic+text combinations. Horizontal bars connect pairs of systems that are significantly different. Inside the boxes, solid lines correspond to the mean; dashed lines to the median.

5.3.1 First listening test: acoustic-derived context

Results for TTS systems conditioned on acoustic-derived context representations are shown in Figure 5.6 (a), corresponding to the systems listed in the upper 4 rows of Table 5.1. Only the system conditioned on contextual representations extracted with Deep Spectrum at the utterance level (condition 3) was significantly better than the baseline. The fact that the only system significantly better than the baseline in this case uses a large pre-trained encoder supports **H3**. Although not significant, all other systems conditioned on acoustic derived context representations resulted in slightly higher scores than the baseline, with utterance-level representations tending to be better than word-level ones, agreeing with **H4**.

5.3.2 Second listening test: text-derived context

Results for TTS systems conditioned on text-derived context representations are shown in Figure 5.6 (b), the systems listed in the lower 4 rows of Table 5.1. All models using text derived context representations were significantly more natural than the baseline. These results do not show a preference between extracting context representations from text using either large pre-trained encoder or jointly-trained ones. Although not significantly different between each other, word-level representations tended to lead to slightly higher naturalness than utterance-level ones, agreeing with **H4**.

5.3.3 Third listening test: best models and combinations

We compared the most effective condition using acoustic context (**DS-utt**), the most effective condition using text context (**Phone-word**, which had the highest mean), and two combinations of acoustic and textual derived context: **DS-utt + Phone-word** and **DS-utt + BERT-word**. Acoustic representations obtained with Deep Spectrum were clearly the most effective among the context representations based on acoustic features. Since there was no significant difference between the models

derived from textual context, we included both **Phone-word** and **BERT-word**. Results are shown in Figure 5.6 (c).

The system combining **DS-utt + BERT-word** was significantly better than baseline. It is not surprising that, although in the previous tests **DS-utt** and **Phone-word** were significantly better than baseline, they were not for this one. MUSHRA ratings are relative, with an element of ranking, such that a different set of systems under comparison can lead to a different rating space (especially if the least natural system changes; there is no anchor in our tests).

5.3.4 Discussion

Regarding the main goal of our experiments in this chapters, the results support the general claim that how we represent context matters: a different combination of the design choices represented in Figure 5.1 can lead to significantly better results.

With respect to our hypotheses, we see that, following **H1**, for every listening test, the baseline was outperformed by at least one model conditioned on a representation of context. Supporting **H2**, the best system was obtained by combining context representations derived from both textual and acoustic features. Regarding **H3**, acoustic-derived context representations were significantly better than baseline only when obtained through a large pre-trained encoder. In contrast, textual features seem to provide relevant information both with jointly-trained and pre-trained encoders. Finally, results from the first listening test and the best model in the final test partially support **H4**, considering that utterance-level context representations were better suited when obtained through acoustic features. It seems that relevant context representations can be extracted from textual features both at the utterance and at the word-level. These last two observations could be pointing to the fact that it might be easier to extract relevant information from textual, than from acoustic features, to represent context.

Informally, we observed that conditioning TTS on contextual representations affected the speech in different ways: prosody, pauses, and pronunciation, with the most noticeable changes being prosodic in nature (see next section for more detail). Although we did not ask participants to directly judge prosody, it seems likely that they are implicitly doing so. At the end of each listening test, we included an optional comment box. Several participants mentioned how it was "interesting" or "challenging" to distinguish the different "inflections" in the samples.

5.4 Analysis

5.4.1 Qualitative analysis

Our results indicate that local linguistic context is informative for TTS, providing additional evidence to our results in Chapter 4. Considering our own perception of the synthesized speech and the comments from the listening test participants, the information contained in the context representations leads the model to generate samples with a different prosody (from the baseline and/or when using a different context representation). We propose a small qualitative analysis to understand more how the different context representations are affecting the synthetic speech prosody. Similar to the analysis we conducted in Section 4.5, we synthesize the test utterance

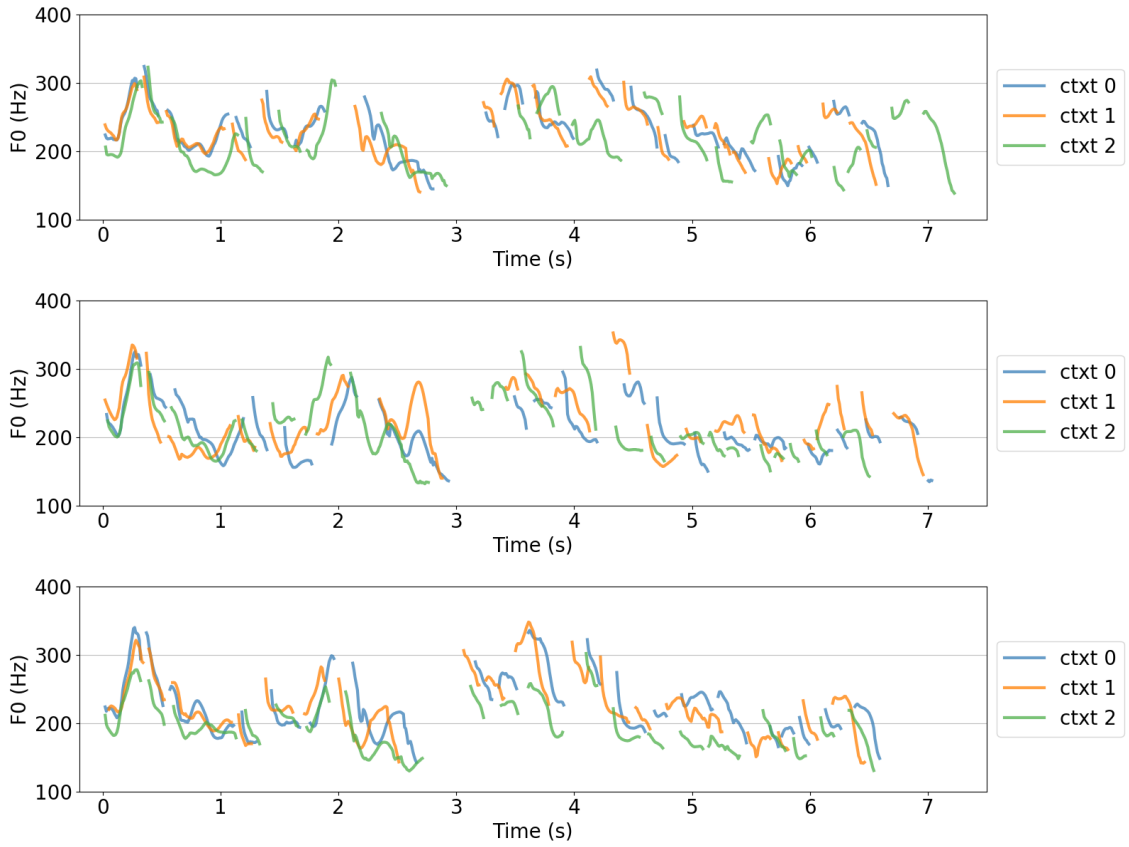

Figure 5.7: Illustration of the effect of context for a single sentence synthesized, from top to bottom, with (a) **DS-utt**; (b) **BERT-word**; (c) **DS-utt + BERT-word**. In each plot, the three f0 contours are the result of using three different context utterances (the same three across all plots).

conditioning on three different context utterances, randomly selected. If context is in any way informative, then the versions of the target utterance will not be identical.

Figure 5.7 shows the resulting fundamental frequency contours when using three context utterances to synthesize the same target utterance. All utterances were the same for all three plots.

In (a), the system is **DS-utt**: it can be seen that while the overall f0 pattern tends to remain similar, there is a global change when using different contexts, shifting the fundamental frequency range: the green contour has a noticeable lower pitch near second 1. It also affects speech rate: notice how the green contour ends closer to 7 seconds while the other two do it near 6.5 seconds.

In contrast, (b) is for the **BERT-word** system: it can be seen how the three different contours differ in local prominence. For example, near 3 seconds, before the pause, the orange contour has the highest peak, while the blue also presents prominence but with a lower strength, and finally the green contour, slightly earlier, seems to be in-between. Notice that this same time location in the first plot was basically identical for the three contexts.

Finally, the third plot (c), is for the combination of both, system **DS-utt + BERT-word**, the plot shows, for example, how the green contour is still overall lower in pitch, and how it differs in local prominence with the other two contours in the peak right before second 4. We can see that nevertheless the effects interact,

as the speech rate differences that were present in the first plot are no longer there (the three contours have a very similar duration).

We can interpret this analysis as that the utterance-level context representations derived from acoustic features tend to produce global effects in the prosody of the synthetic speech, while the word-level context representations derived from textual features affect prosody locally. Considering this, it seems reasonable that the best model was a combination of both. Moreover, consider our similar analysis in Section 4.5: our best model for those experiments, the one using a multi-tasking setting, produced local prosody changes together with global ones, in contrast to the second best model that affected prosody only globally. In contrast to the analysis in Section 4.5, in this section we use fewer renditions here to facilitate the comparison.

Finally, taking into account that the three plots were created by synthesizing the same test and conditioned on the same three context utterances but through different context representations, shows how different the information provided from the context can be depending on the method used to obtain a representation of it.

5.4.2 Synthesizing with synthetic acoustic context

In all the listening tests in Chapter 4 and in this chapter, when testing the use of context representations extracted from acoustic features, we had extracted them from ground-truth samples corresponding to the previous utterance in the data. In a real-world use case, either when synthesizing monologue or dialog data, only *synthesized* previous utterances will be available. In this case, the inference will have to proceed as in Figure 5.8. When synthesizing, for example, a paragraph split into utterances, for each target utterance the process will be run. In a specific run when synthesizing target utterance N, the target utterance synthesized in the previous run, N - 1, becomes the context utterance for the target utterance N, through a feed-back loop. To start the process, for the very first utterance we extract the context representation from a silence clip.

Although we have not tested yet the use of synthesized acoustic context through a formal listening test, we have informally tested the method in Figure 5.8. Intuitively, we think that the use of a synthetic speech as context that might contain errors or lack expressivity compared to using the ground-truth utterance, and through multiple repetitions of the method, could cause a ripple-effect, degrading naturalness at each iteration, breaking the effectiveness of the method proposed here.

When synthesizing chapters of LJ Speech with **DS-utt** and **DS-utt + BERT-word** following the process in Figure 5.8, and through informal listening, we see some degradation in quality with respect to the use of ground-truth context. However, the synthetic speech is still intelligible and the feed-back loop does not degenerate the generated speech completely. We hypothesize that the method is still able to produce speech due to multiple factors: (1) our systems are based on FastPitch, which is a very stable model that rarely produces "failed" sentences ("failed" as attention-based TTS failure modes can produce); (2) when combining context representations from both text and acoustic features, the textual context might act as an anchor; (3) when combining context representations from both text and acoustic features, as we will show in the next section, the effect of the acoustic features seems to be more subtle than when a system relies on context representations derived from acoustics alone.

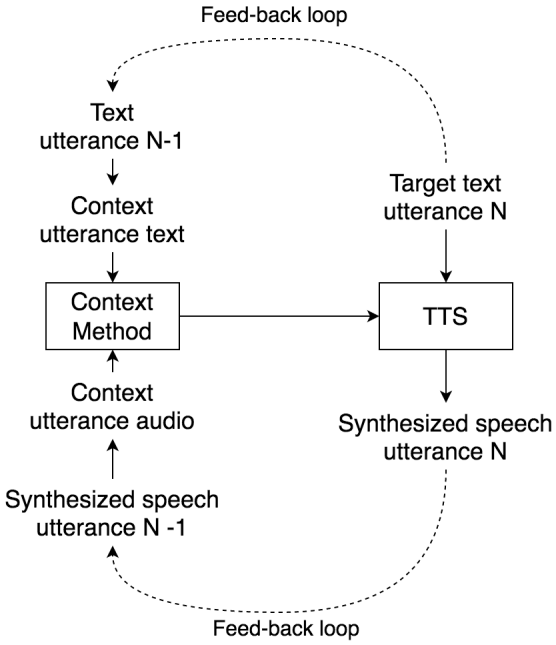

Figure 5.8: Synthesis procedure when extracting context representations from synthesized speech as acoustic context.

Notice though that at this point we have only tried this informally: it is possible that the differences resulting of using synthetic acoustic context might affect the judgements of the participants for our method in a listening test. We will explore this further in Chapters 7 and 8.

5.4.3 The role of utterance-level acoustic context

In this section we analyse further the role of context. In this case, we study only context representations derived from acoustics, and make use of a different method to the one in the previous section. We propose to leverage the synthesis method illustrated in Figure 5.8, where synthetic speech is taken as the context utterance. Through acoustic manipulation we will see if we can impose certain prosodic trends through the use of acoustic context. This can help us understand if a certain prosodic attribute in the context utterance has a direct effect in the target utterance.

For each iteration of the synthesis process, we manipulate the synthesized speech that was produced in the previous iteration, generating a manipulated version of the context utterance. The inclusion of the manipulation step in the synthesis process is illustrated in Figure 5.9. We tested fundamental frequency and duration manipulation. Both were implemented automatically using Praat for Python (Jadoul et al., 2018), which performs manipulation through the PSOLA algorithm.

5.4.3.1 Fundamental frequency manipulation

We want to see if, through manipulation, we can find a direct relationship between the fundamental frequency characteristics of the context utterance and the effects generated in the target utterance, e.g. if the context utterance's fundamental frequency is high, will the target utterance's fundamental frequency also be higher? (and vice versa). In total, we synthesized 4 held-out chapters that amount to about

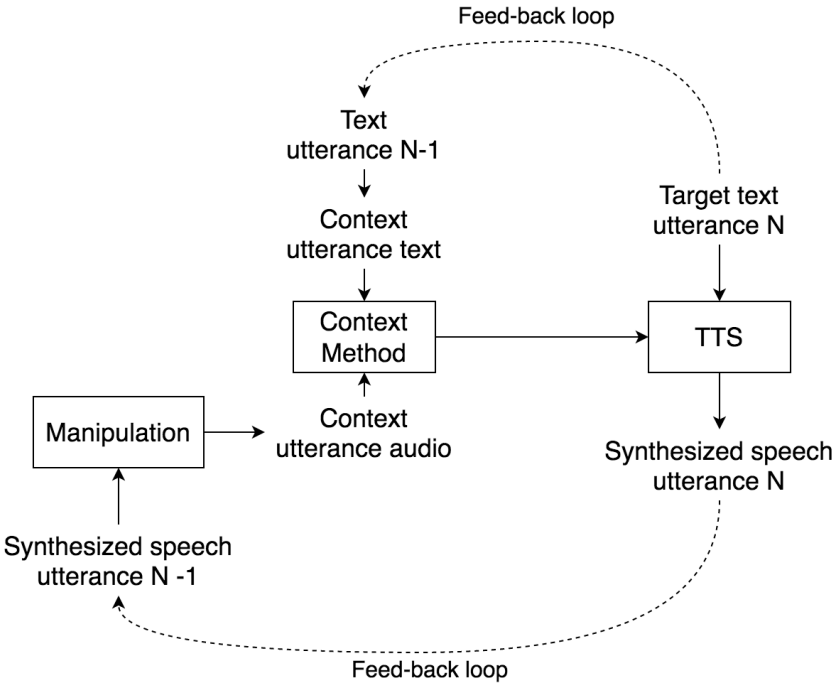

Figure 5.9: Same synthesis procedure illustrated in Figure 5.8, although here, including a manipulation step applied to the acoustics of the context utterance.

600 utterances. Notice that manipulation is performed for every iteration. We compare three different synthesis sets: "**base**", which is normal synthesis without applying manipulation; "***1.2**", where we multiply the context utterance's fundamental frequency by a factor of 1.2; and "***0.8**", where we multiply the context utterance's fundamental frequency by a factor of 0.8. We chose these factors during preliminary trials, as we considered that the resulting speech still sounded natural. We repeat these three synthesis sets for different models.

We visualize the fundamental frequency distributions for each set of synthesized speech, in Figure 5.10 for four different models: (a) **DS-utt + BERT-word**; (b) **DS-utt**; (c) using delexicalised acoustic context only at the utterance-level **Delexicalised-utt** and (d) using delexicalised acoustic context at the utterance-level together with BERT-word, **Delexicalised-utt + BERT-word**. Vertical bars show the mean. These two last models, using delexicalised speech as an acoustic feature to extract the context representation, was trained especially for this analysis, after seeing the results of the first two models.

As the plots show, the manipulation of the context utterance's fundamental frequency, changes the mean towards the used factor only minimally for **DS-utt + BERT-word**. We can interpret this as showing a very subtle trend or relationship between the fundamental frequency of the context and target utterances. However, we can see that for the top-right plot, **DS-utt** not using textual context features, the trend is slightly more noticeable. Noticing this difference, we thought that we could encourage the trend further by removing any segmental information from the acoustics of the context to force it to provide suprasegmental information only, hence the use of delexicalised speech. The two bottom plots show, how by doing this, the trend is more noticeable: the mean of the distribution for the target utterance changes towards the factor used to manipulate the context utterance. We can see

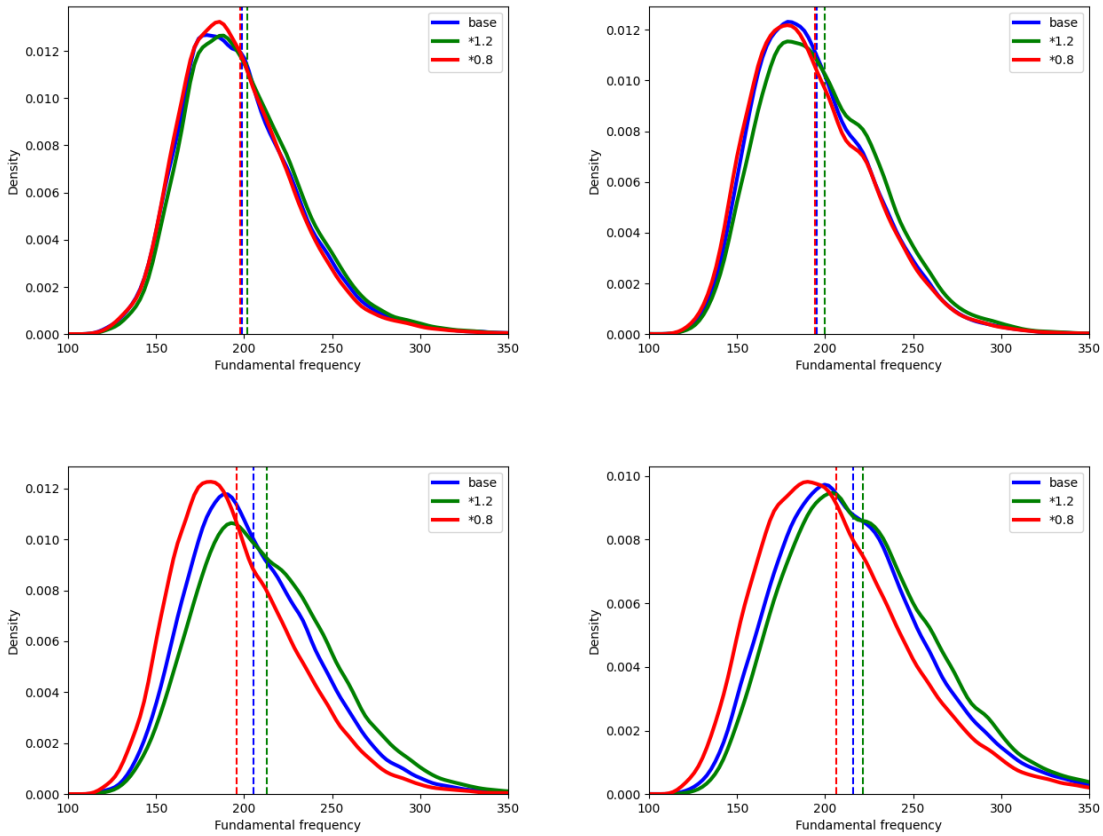

Figure 5.10: Fundamental frequency distribution plots for each set of synthesized speech. We compare the behaviour of four systems: (a) top-left, **DS-utt + BERT-word**; (b) top-right, **DS-utt**; (c) bottom-left, **Delexicalised-utt**, and; (d) bottom-right, **Delexicalised-utt + BERT-word**. Blue distributions correspond to baseline synthesis, without manipulation. Green distributions correspond to synthesized speech, manipulating the context utterance multiplying fundamental frequency by a factor of 1.2, and red distributions correspond to synthesized speech, manipulating the context utterance multiplying fundamental frequency by a factor of 0.8

that the trend is more clear for the bottom-left (in comparison to the bottom-right) plot, again the system using acoustic context only, **Delexicalised-utt**. We think that these differences point to when a model uses both acoustic and textual context, the textual context might have a higher weight for the predictions. The doubt remains if this is due to the source of the features or the difference between utterance and word-level representations, where possibly the model reacts more to a non-constant representation.

This analysis might be showing that, while there is a relationship between the fundamental frequency between context and target utterance, it might be not too strong. In seems that the relationship between the acoustic representations of the context and target utterances might be more complex than a particular acoustic value such as fundamental frequency. Finally, the distributions for each system are not significantly different: the plots are provided only as a way to visualize the trends.

5.4.3.2 Duration manipulation

We applied the same method described above but this time manipulating duration. We were seeking evidence that speech rate could have a direct relationship between context and target utterance, e.g. if the context utterance's speech rate is high, will the target utterance's speech rate also be higher? (and vice versa).

We tested two types of duration manipulation: global (as before, multiplying by a factor), and local, by randomly selecting silences or words (following alignments) from the context utterance and modifying their duration. We ran these two types of manipulation for the same four systems presented above, obtaining again, three set of synthesized utterances per system. Despite the manipulation, all the synthesized sets were identical to each other for a given system: the duration manipulation did not have any effect on the synthesis of the target utterances.

We think there could be several complementary explanations for these results: (1) the Deep Spectrum embeddings do not capture speech rate/duration; (2) there is not a direct relationship between context and target utterances; (3) there is not a direct relationship between context and target utterances, but specifically for this speaker only; and (4) duration changes are entangled with other acoustic attributes. We think that these results confirm the complex relationship between the context utterance's acoustic characteristics and the target utterance.

5.4.4 The role of word-level textual context

As described in Section 5.2.2.2, we proposed a new Context Method to inject word-level Contextual Representations into a TTS architecture. Although we based our design on the methods in G. Xu et al. (2020), we decided to implement a simpler method that could potentially be interpreted more easily. Given the design of our method, where attention is used to calculate the relevance between word-level context and word-level target representations, we looked for any patterns in the attention weights for the word-tokens from BERT that were selected as relevant. In general, we could not find any patterns. This was also the case for Tiedemann (2017)'s similar approach for NMT. In Chapter 6 we will consider the extraction of more interpretable features from textual context to improve TTS systems, in a way more similar to the supervised methods mentioned in Section 5.1.

5.5 Conclusions

In this chapter, we focused on testing context representations. Our experimental results showed that different ways of representing context, used to condition TTS systems, can significantly change the synthesized speech quality. Some representations bring improvements that make the systems better than the baseline. In particular, we saw that the combination of contextual representations derived from acoustic and textual features from the context utterance yields the best results. Acoustic-derived context representations are more effective when at the utterance-level, while in general, text derived context representations are more effective at the word-level. Finally, as a trend, obtaining contextual representations through large pre-trained encoders yields better results than using encoders jointly trained with the TTS.

Even if context is not used explicitly to improve prosody, this seems to be the aspect that is affected the most. The analysis we carried out subsequent to the experimental results provided further insights into the use of context. We saw that utterance-level contextual representations derived from the context's acoustics affect the prosody of the target utterance globally. In contrast, word-level contextual representations derived from the context's text affect the prosody of the target utterance locally. When using synthesized speech as context for inference of multi-sentence speech such as audiobook chapters, and applying manipulation of the context's acoustics, we found that there is a subtle relationship between the context utterance's fundamental frequency and the target utterance's fundamental frequency. We did not see the same in terms of duration, and we could not find patterns in the word-level text-derived context representations.

5.6 Chapter contribution

The experimental results in this chapter provide further evidence to confirm that we can improve synthetic speech naturalness by conditioning TTS systems on context representations, answering **RQ1** from the Introduction chapter. We tested the use of context representations derived from text for the first time in this thesis, in line with related work by G. Xu et al. (2020) and Guo et al. (2021). To our knowledge, we are the first of carrying on a systematic comparison between context representations to condition TTS systems.

The analysis we presented gave us deeper insights into the role of context, especially regarding the use of acoustic features from the context. From the experimental results, we could establish a best Context Method for the time being (**DS-utt + BERT-word**), which we will explore further in future chapters. In Chapter 6 we will look further into how to extract textual features from context that could be more interpretable.

While in the previous chapter our best system required the use of an additional loss, we do not apply the multi-task framework here. This is convenient considering that FastPitch already works with three different losses (mel spectrogram, log duration, fundamental frequency), and therefore we preferred not to use an additional loss in the experiments. Moreover, the number of systems to compare would have increased even further. We believe that considering the analyses presented in both chapters, the use of word-level textual context is producing the localized prosodic effects we saw were the main benefit of using an additional loss in the previous chapter.

Finally, we addressed some of the points we raised at the end of Chapter 4: considering the arguments in "TTS architecture" we switched from Tacotron 2 to FastPitch, which is a more reliable base architecture, and the experiments in this chapter were designed to addressed the issue of the "Context representation and conditioning method". Other issues raised in Section 4.6 were only superficially addressed here ("The "first sentence" and synthesized acoustic context"), while the rest have not yet been explored. All of these will be considered in subsequent chapters.

Part II

Additional methods

Chapter 6

Alternative approaches to incorporate context

6.1 Introduction

This chapter presents the experiments and results carried out during my internship with Amazon Alexa UK. The internship lasted six months, in a part-time fashion, remotely, during the end of my third year of PhD. Although the team I worked with and I had a common interest in the use of context for TTS, the experiments that will be presented in this chapter were designed to fit both theirs and my past work. Some details and information regarding the experiments will be omitted for confidentiality reasons.

While the use of context is still the overarching topic of this chapter, in common with the thesis, unlike in any other chapter, we make use of conversational data. Instead of continuing building on the method proposed in the previous chapters, we test alternative approaches to leverage context for TTS. It is important to note that the scope of the experiments was limited by the time frame of the internship, which not only included the experimental work but also data collection and data pre-processing.

6.1.1 Podcast data and conversational Text-to-Speech

Conversational TTS would probably benefit from using spontaneous and authentic conversational data to achieve a high level of naturalness when embedded into conversational applications. Unfortunately, eliciting natural and expressive conversational data in a studio is difficult even from the best actors, therefore, making the use of found conversations desirable. Podcasts have been proposed by Székely, Henter, and Gustafson (2019); Székely, Henter, Beskow, and Gustafson (2019); Yan et al. (2021) as a rich source for such data, although they are not yet extensively used for TTS. Even if podcasts have many advantages as a source of data, they also have many of the challenges typically encountered when using found data. Some characteristics that make podcasts a good data source are:

- Podcasts are increasingly popular, with millions of hours of audio available online.
- There is a diversity of speakers, including amateur and professional ones.

- There is variety in the number of speakers that interact in a podcast, from single speaker to multi-party conversations.
- Some podcasts have been "on air" for a long time with permanent speakers/hosts.

All these characteristics make them an exciting source of rich data for TTS. However, at the same time, collecting podcasts to use as training data for TTS suffers from the general challenges of using found data (Baljekar & Black, 2016). As identified by other authors, and relevant for the podcast data, some challenges that we encountered were:

- There are no ground-truth utterance boundaries for the data. In order to segment the data for TTS training we will need to make some assumptions and segment the data into suitable audio-text pairs.
- Overlapping speech is not handled by TTS state-of-the-art architectures.
- Potentially, multiple non-speech audio phenomena can be present, such as laughter, clapping, music, excerpts from other media (such as other videos), etc.
- Regarding the spontaneous nature of most of the shows, spontaneous speech phenomena will potentially be present, such as filled pauses, hesitations, false starts, etc.

These need to be coped with somehow either during pre-processing of the data or during TTS training. A limited amount of previous work (Székely, Henter, & Gustafson, 2019, 2019; Yan et al., 2021) has made successful use of podcast data for TTS, although to our understanding, it all trains with the same podcast show, extracting data from a single speaker only.

Székely, Henter, and Gustafson (2019) collected data from one speaker only from a conversational podcast. They use a speaker-dependent breath detector to segment the data into utterances. This gave better results in comparison with using voice activity detection algorithms, however, the breath detector requires a seed of manual annotations to train it. The audio was transcribed automatically, including transcriptions for filled pauses. Next, Székely, Henter, Beskow, and Gustafson (2019) trained Tacotron 2 models with 9 hours of collected data, fine-tuning from a base model trained on read speech.

Yan et al. (2021) trained AdaSpeech 3 by fine-tuning a base model trained on read speech to spontaneous speech from the same podcast used by Székely, Henter, Beskow, and Gustafson (2019), although in this case, collecting 28 hours of data. They automatically transcribed the data including filled pauses, and they segmented speech in fragments of 7 to 10 seconds. The architecture is especially designed for spontaneous speech with a module to predict location of filled pauses and a duration predictor based on a mixture of experts to model three different rhythms (fast, medium, slow). They find the model is significantly better than AdaSpeech 1.

When looking for work more closely related to the one presented in Part I of this thesis, in combination with conversational data, we found Cong et al. (2021) and Guo et al. (2021), where both leverage context to condition TTS models when training

with conversational data. All other papers found at the time are making use of single speaker data and in the form of audiobooks (including our own previous work). Guo et al. (2021) (described in Section 5.1.1) trains with Chinese conversational data a Conversational Context Encoder together with their Tacotron 2-like TTS. The input to this encoder is a dialog history in the form of sentence embeddings from BERT, where each has a label for the speaker the utterance belongs to. They use a data set of 3 hours and therefore they start from a pre-trained model. Cong et al. (2021) uses the acoustic context method we proposed in Oplustil-Gallegos et al. (2020) to model speaker entraintment for two-party dialog data. Cong et al. (2021) trained with 7 hours of data. Both of these papers leverage conversational data especially recorded for their research.

6.1.2 Research questions

In this chapter we focus on using dialogue speech extracted from podcasts to train TTS systems, while testing if these systems improved when leveraging contextual information. We have two research questions:

- **RQ6.1**: Can we train stable and intelligible baseline TTS systems with the collected podcast data?
- **RQ6.2**: Can we improve the synthesized speech obtained with the baselines by leveraging contextual information through alternative methods to the ones presented in Part I of this thesis?

To achieve this, in Section 6.2 we start by looking for a two-party dialogue podcast and describing the processing of the data. Then, we present 4 sets of experiments: Section 6.3 and Section 6.5 attempt to train suitable baselines to answer the first research question, while Section 6.4 and Section 6.6 consider the use of context in two distinct, non-related approaches.

In Section 6.4 we experiment with the use of contextual information to improve the selection of reference utterances at inference time to synthesize with a TTS+VAE system. In Section 6.6, following the conclusions for the use of textual representations from context in Section 5.4.4, we look into extracting interpretable features from textual context to augment the TTS input and improve the synthetic speech naturalness.

Notice that in this chapter, terminology wise, we refer to *podcast* as a series and use it interchangeably with *show*, where both are a collection of *episodes*.

6.2 Data collection and processing

We searched and collected a new podcast instead of using some of the resources already available. We did not want to use the same podcast as Yan et al. (2021) and Székely, Henter, Beskow, and Gustafson (2019), in order to extend the number of shows that have been used for TTS training. Although recently Spotify had released a data set of podcasts (Clifton et al., 2020), Amazon's request to use the data was never answered. One other available data set was the MSP-Podcast corpus (Martinez-Lucas, Abdelwahab, & Busso, 2020) which has 100 hours of speaking

turns from podcasts collected for the field of emotion recognition. Unfortunately this data set does not contain full episodes but instead random segments from many podcasts, which decreases the amount of data per unique speaker. However, we followed a very similar data processing approach as they did.

We looked for a podcast with a highly natural and expressive conversational style. Recording quality should be good and consistent throughout the different episodes. We avoided interview podcasts: the dynamics of the interview can be quite different to a normal conversation, as it is likely that the interviewer will ask relatively shorter questions in comparison to the answers of the other speaker. Instead, we looked for podcasts with 2 permanent co-hosts of equal importance. We preferred relatively proficient speakers, that had been hosting the show for a considerable time in order to have many episodes -and therefore hours of speech- available for training. We avoided podcasts with permanent background music.

We started by exploring the metadata available in the data.world website[1] portal containing metadata information about podcasts online, listing 74820 shows. To short-list the potential candidates, we started by keeping only the shows identified as US English ($\sim$ 17k), which was the majority. We calculated the amount of audio per show and selected the 20 podcasts with the most data.

We manually checked the 20 podcasts short-listed. From the links that were still working, we selected 2 shows that fit the required characteristics. Given Amazon's confidentiality requirements we cannot disclose the identity of the podcasts in this thesis, but each one has 2 female hosts from the US. From here on we will refer to them as **Podcast 1** and **Podcast 2**.

Amazon dealt with all the necessary actions to take in order to use the data legally and ethically, including compensation for the speakers. Amazon was in charge of this without participation of the University of Edinburgh as they became the owner of the data: the data was only stored on Amazon-owned servers, and it was only accessible to other Amazon employees. Moreover, the synthetic speech generated with the models trained with this data was only presented to other Amazon employees. I did not keep samples of the synthesized speech or the ground-truth data. We have also ensured that the information presented in this chapter is insufficient to identify the speakers, or the names of the podcasts.

All the episodes available in the RSS feed were automatically downloaded. Using the available episode description, we selected only the episodes where the 2 latest co-hosts were present, which were 99 for Podcast 1 ($\sim$ 26 hours) and 194 for Podcast 2 ($\sim$ 115 hours). In the subsequent sections we will describe every step taken to process the collected data into a suitable data set for TTS for the experiments in this chapter. Additionally, we performed several steps to analyze the data in order to filter it: the output of this analysis was retained as metadata that we leverage further in our experiments.

6.2.1 Automatic transcription and utterance segmentation

The collected data was transcribed per episode, using the AWS Transcribe API[2], which provides text transcription and speaker diarization. The API requires specifying maximum number of speakers in the sample. Considering that even though

[1]https://data.world/brandon-telle/podcasts-data set
[2]https://aws.amazon.com/transcribe/

the shows were mostly not interviews, some episodes, nevertheless, had guests, so we used a value of 5.

All episodes had advertisements about other podcasts at the beginning and at the end. To avoid these interfering with the transcription and diarization, we automatically cut the starts and ends of all episodes.

By inspecting the number of speakers identified by AWS Transcribe, we filtered the episodes that could potentially have other speakers besides the two main hosts. To filter out or keep an episode we considered:

- Keep all episodes with only 2 speakers identified.
- Discard all episodes with only one speaker identified.
- If an episode had more than 2 speakers identified: if 2 of them sum up most of the data (90%) and the data is roughly evenly distributed between speakers (a difference less than a 20%), keep it.
- Otherwise, if the audio is distributed more evenly through all speakers, discard it (e.g. there is a potential guest).

By applying this procedure, we kept 61 episodes, ∼12.5 hours for Podcast 1 and 123 episodes, ∼54.7 hours for Podcast 2. Notice that these amounts are fairly distributed between the two speakers (for example, for Podcast 1 there are about 6 hours of speech per speaker).

From the collected speech we split the episodes into utterances using the AWS Transcribe output alignments. We follow the procedure:

- Split when there is a change of speaker.
- If the resulting segment is longer than 10 seconds, split where there is a punctuation symbol (with a minimum duration of 0.5 seconds, potentially associated with a pause).
- If the resulting segment is still longer than 15 seconds, split where the longest silence is.

The resulting number of utterances were 11040 for Podcast 1, and 54845 for Podcast 2. For both podcasts, most of the resulting utterances last between 1 to 4 seconds.

6.2.2 Overlapping speech analysis and assigning speaker labels

Although the AWS Transcribe output provides speaker diarization, it does not provide information about overlapping speech, which it is fundamental to detect for our experiments. State-of-the-art TTS does not handle overlapping speech and therefore we need to detect which utterances contain overlap.

We ran every utterance through a model from the Pyannote library (Bredin et al., 2020) for audio analysis, which came pre-trained on the AMI corpus (McCowan et al., 2005). We obtain a percentage of overlap per utterance, which we store as metadata. Figure 6.1 shows the distribution per utterance.

In terms of total duration, Table 6.1 shows the percentage of overlapping speech in four ranges. We can see that most of the data for each podcast contains no overlap according to the model. When inspecting the audio, we observed that, although this is in general correct, very short overlapping speech falls into this category. For example, it is very common that while one of the speakers is talking the other will employ back-channels to show she is listening to her, such as "yeah", "right", "uh-huh".

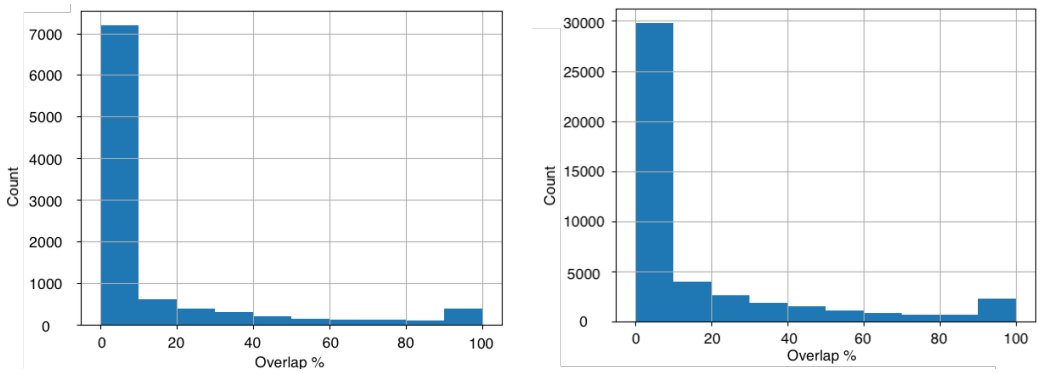

Figure 6.1: Histograms of speaker overlapping percentage per utterance. Podcast 1 (left) and Podcast 2 (right).

	Podcast 1	**Podcast 2**
No overlap	∼7 hrs	∼26.4 hrs
0% > 10%	∼1 hr	∼6.1 hrs
10% > 50%	∼2.5 hrs	∼17.3 hrs
More than 50%	∼1hr	∼5.1 hrs

Table 6.1: Percentage of overlapping speech in terms of hours per podcast.

The AWS Transcribe API assigns numbered speaker labels given the order of appearance in the audio. For example, if one of the speakers starts first in one episode, but not in the next one, she will not have the same number assigned. Therefore, we need to make sure that the same speakers across episodes are assigned the same labels. We extracted speaker embeddings for every utterance in the data with Amazon's internal models and ran different stages of clustering with K-means. Clustering will also help us to find overlapping speech and any other type of outlier utterances (laughter, music, other speakers, etc.).

First, we made a general pass clustering with K=2 the utterances from each podcast (show) separately, and we analyzed the samples that are further away from the centroids. We inspected these samples and found episodes with uncommon recording conditions or with guest speakers. We discarded these episodes from the data.

Figure 6.2 shows the speaker embeddings per show using t-SNE visualization of the resulting data. We can see that, after removing outliers, the data still does not simply have two clusters, one for each host, but rather there are more clusters forming. Next, we follow the assumption that there should be two dominant speakers, the hosts, corresponding to the two largest clusters per show. We perform clustering again, looking for the number of clusters that will allow us to split most of the data

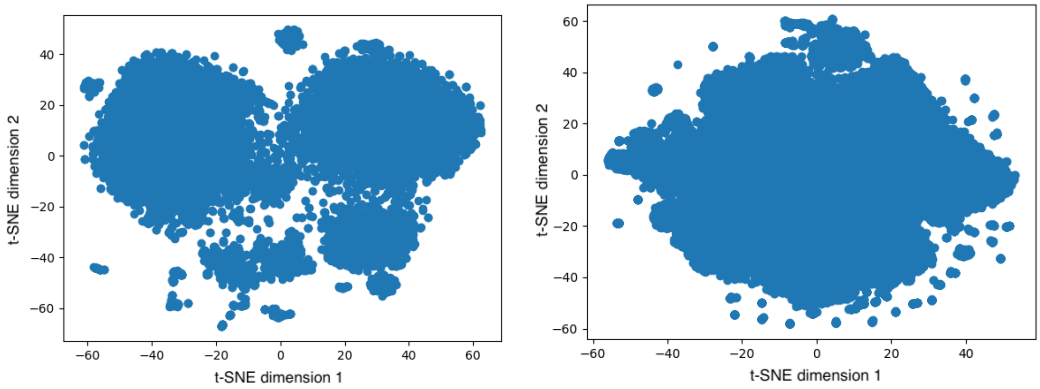

Figure 6.2: t-SNE for speaker embeddings after general first outlier removal. Podcast 1 (left) and Podcast 2 (right).

into two main clusters. The utterances that are far from the centroid of one of the two dominant clusters can be consider suspicious: corresponding to other speakers or non-speech. By analysing the distance to the centroids of the larger clusters, we assign a confidence level for every utterance (from 0 to 2, the highest):

- If the utterance is closest to a centroid that is not one of the two dominant ones for that episode, the confidence level is 0.
- If the utterance is closest to the dominant clusters, but further than a threshold (in cosine distance), the confidence level is 1.
- If the utterance is closest to the dominant clusters, but closer than the threshold, the confidence level is 2.

We store the confidence level per utterance as metadata. We use K=3 and a threshold of 0.6 for Podcast 1 and K=8 and a threshold of 0.8 for Podcast 2. Table 6.2 shows the number of utterances per level for the two podcasts. Finally, we assign a speaker label for each utterance considering if it is closest to the first or second larger cluster when clustering all episodes per show together. If we visualize again with t-SNE the two shows (Figure 6.3), but only those utterances with confidence level 2, we can see that there is a better separation between two dominant clusters.

Confidence	**Podcast 1**	**Podcast 2**
Level 0	1570	11416
Level 1	2186	5145
Level 2	5861	28936

Table 6.2: Utterances per speaker label confidence levels per show.

6.2.3 Word Matching Rate analysis

To further enrich our analysis of the data, we ran it through a second STT model, internal to the Amazon TTS team, and based on a different system to AWS Transcribe.

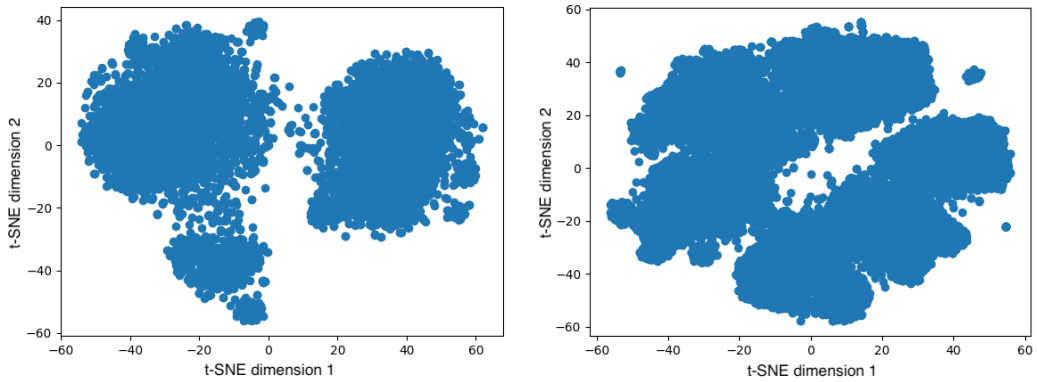

Figure 6.3: t-SNE for speaker embeddings of the utterances with highest confidence of belonging to one of the two hosts. Podcast 1 (left) and Podcast 2 (right).

We obtained Word Matching Rates (WMR[3]) for every utterance, with respect to the transcriptions obtained with AWS Transcribe. Figure 6.4 shows the distribution of WMR per utterance. On average, there is a 12.06 % WMR for Podcast 1 and 13.88 % WMR for Podcast 2. These values are stored as metadata.

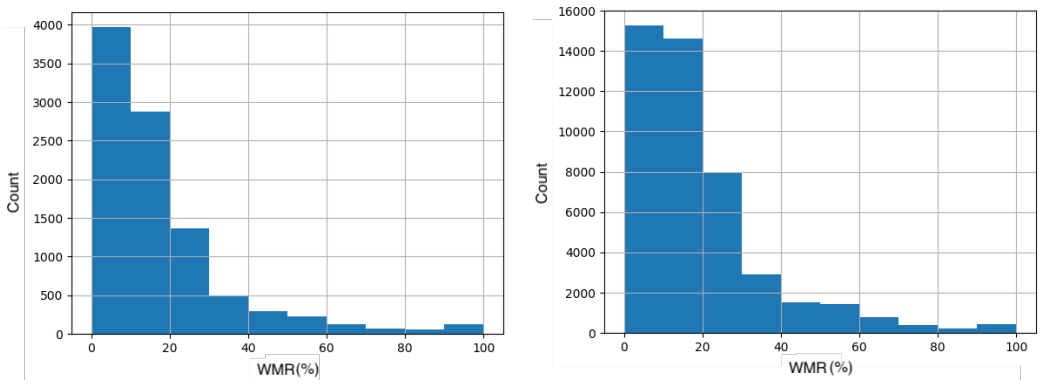

Figure 6.4: WMR distribution per utterance for each podcast. Podcast 1 (left) and Podcast 2 (right).

6.3 Baseline models

In relation to the first research question stated in Section 6.1.2, we started by training baseline models for the collected podcast data. We tested if we could build stable and intelligible systems, using an internal Tacotron 2-like architecture. We trained preliminary models to explore the following experimental conditions regarding the optimal training configuration:

- Flat-start training of the podcast data vs fine-tuning to the podcast data from a base model trained with studio recorded data (female single speaker).

[3]As we do not have ground-truth transcriptions we can't have a Word Error Rate in the conventional sense. When comparing different automatic transcriptions we use the term Word Matching Rate.

- Single-speaker vs multi-speaker models (where multi-speaker groups together the respective speakers from each show, therefore, two models with two speakers each).

- What subset of the data to use, balancing amount of the data vs quality of the data, filtering on metadata values.

Preliminary training was evaluated by informal listening, which was sufficient to draw conclusions as the model differences were evident. Fine-tuning to the podcast data from a base model was found superior, consistent with the results of Székely, Henter, Beskow, and Gustafson (2019) and Yan et al. (2021). The models trained from a flat-start were barely intelligible and very unstable. Multi-speaker training, fine-tuned from the base model, was better than single-speaker, especially for Podcast 2. Finally, a subset of 4 to 5 hours of the best data (as defined by our metadata parameters: overlap lower than 30%, speaker label confidence level >= 1 and WMR lower than 50%) resulted in the most stable and intelligible baselines.

Even though as explained above some models were better than others, in general, the baseline models were not very reliable: attention failures were very common, affecting intelligibility and stability. Notably, Podcast 2 was evidently worse than Podcast 1, despite having more data. Therefore, we decided to use Podcast 1 only for the remaining experiments in this chapter, as it seemed to provide better quality data.

Subsequently, we explored an additional approach to improve the baseline by using data augmentation. From the comparisons in the preliminary training, the choice that had the most impact on the model quality was fine-tuning to the podcast data from a base model trained on studio data (from a different speaker). Therefore, we considered using the approach described by Huybrechts et al. (2021) to augment data and improve synthetic speech quality, as this approach was designed to augment low-resource expressive data across speakers.

The augmentation procedure has two main steps: (1) a voice conversion model is trained, CopyCat (Karlapati et al., 2020), is trained using both the studio and the podcast data; then (2) inference is performed with the model, using as input the studio data to convert it to the speaker identity of the podcast data, where the output is the augmented data. The studio data we used comprises ∼25k utterances. This means that the data augmentation procedure generated an additional ∼25k utterances for each of the two speakers of Podcast 1.

Lastly, we train three baseline models, now including the new augmented data, where we compare three different options to combine the data sets through fine-tuning. The models compared were (where → stands for "fine-tune to", and the podcast data comprises ∼2k utterances per speaker):

- Model 1: studio data → podcast data.

- Model 2: studio data + podcast data → podcast data.

- Model 3: studio data + augmented data → podcast data.

A team of experts informally listened to the synthesized speech of the three systems. Unanimously, Model 3 was considered the most stable and expressive. Interestingly, although Model 3 is quite expressive for a TTS system, it is not nearly

as expressive as the training data. Often the model incorrectly predicts emphasis and speech rate. Attention issues were still present, although in a lower proportion than the other systems.

6.4 Variational Autoencoder inference with contextual information

The first alternative method we propose is to leverage context to improve reference selection for inference with a TTS+VAE system. Such systems use a reference encoder (Skerry-Ryan et al., 2018) whose reference embedding is processed by the VAE. As described previously in Section 4.2.1, the VAE encoder learns a distribution: the mean and the standard deviation of the latent variable z. In order to encourage the learning of a useful latent space, the distribution is forced to be Gaussian through a regularization term in the loss, which is applied to update the parameters of the encoder only. To sample a z vector from the distribution, the reparameterization trick is applied to allow back-propagation to pass through the random sampling step, originally proposed in (Kingma & Welling, 2013). The z vector sampled is used to condition the TTS system (Y.-J. Zhang et al., 2019).

At inference time, a z vector must be obtained either through a reference embedding input to the VAE, or by sampling the latent space directly. For example, a centroid z vector can be calculated by taking the mean of all the z vectors for each training data sample, generating a vector that captures the mid point characteristics of the latent space. Another popular option is to use a reference audio to drive the system. This can be done manually or automatically. In the experiments in this section, we want to test if, by considering context, we can select a better reference than if we do not consider it. This is a similar approach to that in Tyagi et al. (2020).

We identify three sources to find the best reference audio to run inference with: (1) the text to be synthesized; (2) the text of the previous utterance; (3) the audio of the previous utterance. The method proposes to measure the distance of these sources to each available reference candidate in the podcast training data. Then, the closest training sample will be used as the reference. To measure the distance, we extract representations for the utterances. The method is illustrated in Figure 6.5.

Considering the three sources indicated above, we design three variations of the described method:

- **Method 1**: compare the text we are synthesizing to each (isolated) sentence text in the training data. Extract representations for every sentence with BERT.

- **Method 2**: compare the text of the previous sentence and the text we are synthesizing, together, to every pair of sentence texts in the training data. Extract representations for every pair of sentences with BERT.

- **Method 3**: compare the previous sentence and the text we are synthesizing, together, including the audio of the previous utterance, to every pair of

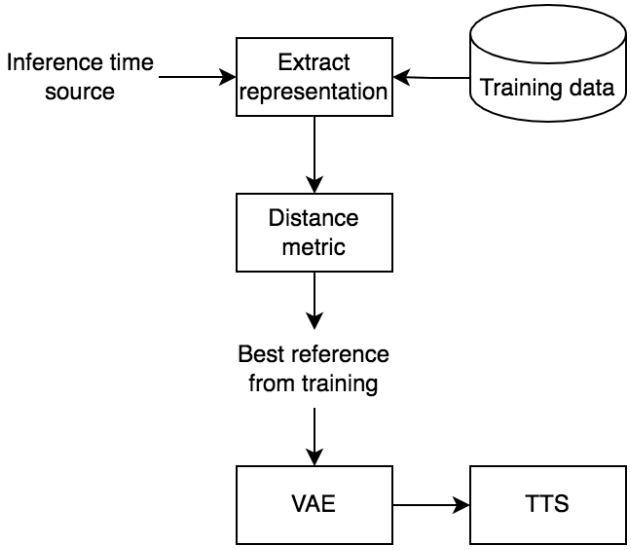

Figure 6.5: Reference selection method.

utterance texts in the training data, including the audio of the previous utterance. Extract representations for every pair using acoustically-shifted BERT (described in the next Section).

We can see that Method 1 is a baseline method where we consider only the text for isolated utterances to find the reference. Method 2 includes the use of context by considering the previous utterance too. Finally, given our experimental results in Chapter 5, we also consider the inclusion of acoustic information when using the context utterance. Figure 6.6 shows the three methods described. We use cosine distance for all methods, which is used to compare the [CLS] vector for all extracted BERT representations. Recall that as explained in Section 2.6.1 the [CLS] token is a summary of the complete sentence.

6.4.1 Acoustically-shifted BERT

Rahman et al. (2020) proposed a method to incorporate multi-modal features into BERT for sentiment analysis. While they include both audio and images additionally to text, in this experiment we incorporate audio only. They propose the use of a Multimodal Adaption Gate (MAG) that shifts the internal representation of BERT with respect to the additional modalities for every text token. The method is applied after embedding the text in BERT, right before the transformer encoder (this location was experimentally found best). No other changes are made to the BERT architecture.

MAG starts with two linear layers. For the first one, the concatenation of the text embedding and the acoustic features are the input, output followed by a ReLU activation. For the second one, the acoustic features only are the input. The two outputs are multiplied, obtaining a vector H, the "non-verbal displacement", as it is called in the paper. A weighted sum of the text embedding and H is calculated by multiplying H by a scaling factor and summing it to the text embedding. The result is run through drop-out and, finally, layer norm is applied.

We propose to use the method of Rahman et al. (2020) to obtain representations that take into account acoustic information to select a reference for VAE+TTS

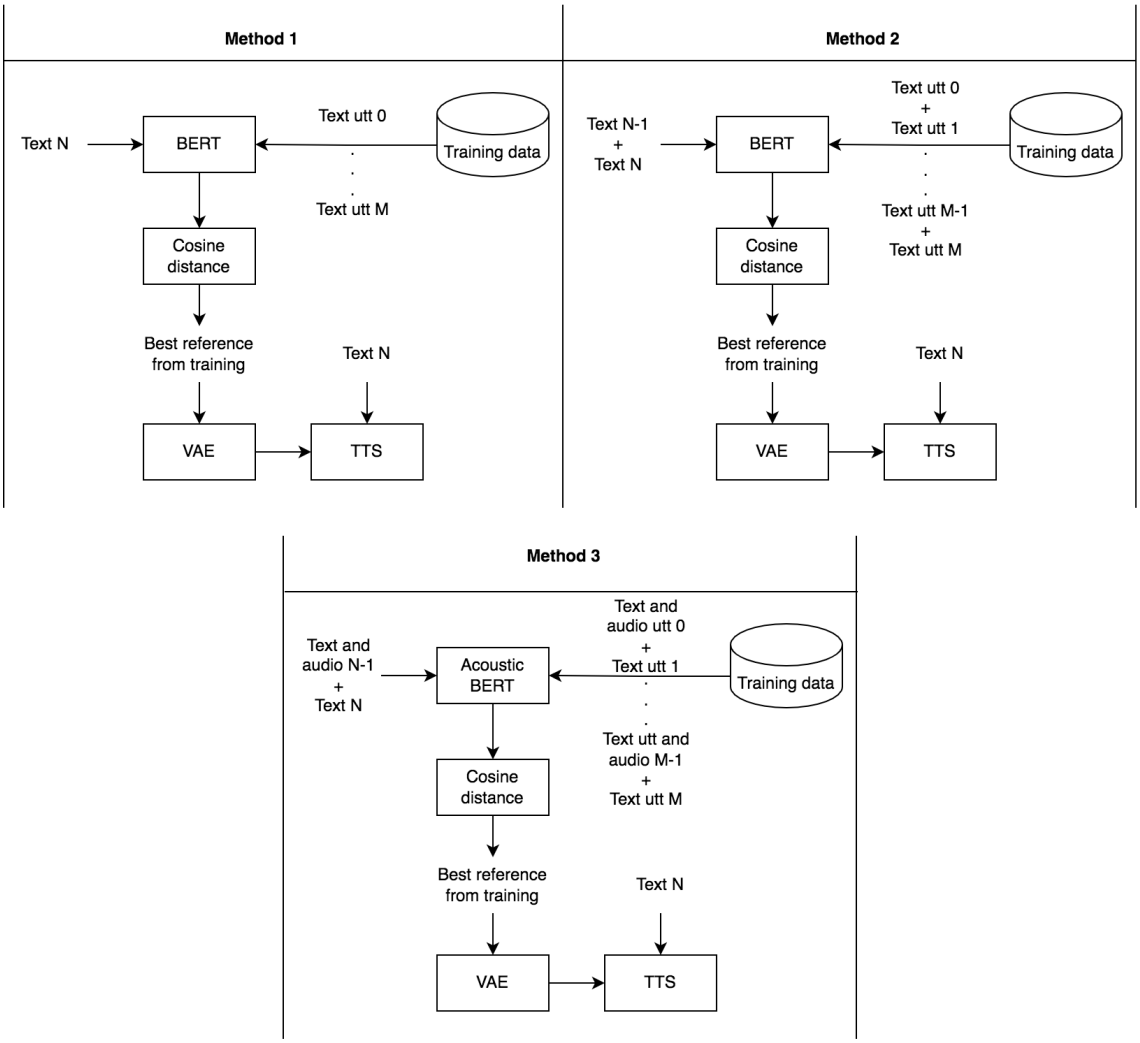

Figure 6.6: Proposed reference selection methods.

inference, when using context (Method 3). As the other two methods make use of BERT too, this implementation allows extraction of the desired representation for Method 3 whilst making minimum changes from the other methods, allowing fair comparison. We made use of the original implementation by the authors[4], modifying it for our experiments.

Rahman et al. (2020) use the COVAREP toolkit to extract acoustic features (such as f0, VUV, glottal source parameters, etc.). The features are forced aligned to the text, using word boundaries to average them at the word level. As COVAREP appears to be no longer maintained, we decided to use the OpenSmile toolkit to extract features, with the "ComParE 2016" feature set (which we found to be the best compared to the other available feature sets).

To apply this implementation to Method 3, we modified some aspects with respect to the original code:

1. Rahman et al. (2020) fine-tuned BERT to a down-stream classification task. We fine-tuned to the original NSP task in BERT (Chapter 2 Section 2.6.1), using all the podcast data.

[4]https://github.com/WasifurRahman/BERT_multimodal_transformer

	WMR (%)	
Inference method	**Speaker 1**	**Speaker 2**
Centroid	8.46	9.79
Method 1	11.39	12.32
Method 2	11.39	13.11
Method 3	10.84	12.24

Table 6.3: WMR comparison for synthesized speech when using the three reference selection methods.

2. As already mentioned, they have multi-modal features, including both audio and video. We used only audio.

3. We adapted the data loader to create sentence pairs for the NSP task, where the acoustic features are only used for the previous utterance (padding to zeros the current one).

As for Method 1 and Method 2, we used vanilla BERT, fine-tuned to the podcast text data to make it comparable to the acoustically-shifted BERT.

6.4.2 Text-to-Speech systems and objective results

For the TTS model, we used the same architecture as the baselines, but this time additionally including a VAE with a reference encoder. The reference encoder has the architecture in Skerry-Ryan et al. (2018), taking as input the mel spectrogram, encoding it with a dimension of 128. This is concatenated to an embedded speaker label. The concatenation is input to the VAE, with and the output is then concatenated to the encoder outputs of the Tacotron 2-like architecture, where the speaker labels are concatenated again. For more details, see the description by the Amazon team in (Strelec, Rohnke, Bonafonte, Łajszczak, & Wood, 2021). We trained the models with the same procedure described in Section 6.3, which was found best for the baseline (using data augmentation and fine-tuning).

To test which of the reference selection methods worked best, we measured WMR of synthesized speech (300 utterances) together with informal listening test by a team of experts. Additionally, we compared inference with the three methods described in Figure 6.6, to running inference using the centroid of the latent space of each speaker. To calculate the centroid, we extract the z vectors for every sample in the training data per each speaker. The WMR comparison can be found in Table 6.3.

We can see that the WMR is the lowest when running inference with the centroid. This is confirmed when listening to the speech, being the most stable and intelligible system, but, the least expressive. All the reference selection methods have a higher WMR. When listening to the samples, we confirm that intelligibility is lower than for inference with the centroid, but with higher expressivity.

However, the problem is that none of the three methods was evidently better than the other, because all of them were extremely inconsistent in their results: sometimes the synthesized speech would have excellent quality and expressivity, while other times it would be extremely poor, both in segmental and suprasegmental quality. Some of these failures could be caused by the attention mechanism, as we

saw for the Tacotron 2-like baseline, but in other cases the reference selection was indeed sub-optimal.

Bearing in mind that other authors had reported good results when using similar methods (Tyagi et al., 2020; Karlapati et al., 2021), we considered two main factors that can explain these results: (1) the BERT vectors are not a good approximation to represent this training data, considering the data's spontaneous characteristics; and (2) the available reference candidates (training data) are too sparse and not sufficient to cover the expressivity space of the speech in the training data, therefore, not containing adequate references for the text to be synthesized (which contrasts with Tyagi et al. (2020); Karlapati et al. (2021), where they use studio data).

	WMR (%)	
Architecture	**Speaker 1**	**Speaker 2**
Tacotron 2-like	11.96	11.68
FastPitch	7.67	8.7

Table 6.4: WMR comparison over synthesized speech (300 samples).

6.5 Improving the baselines

Before continuing with the experiments to incorporate the use of contextual information to improve models trained with the podcast data, we looked into further improving the quality of our baseline models. First, we compare FastPitch to Tacotron 2. We trained a baseline model with the FastPitch architecture, using the exact same training data and features as for the Tacotron 2-like models trained in Section 6.3. We compare WMR for the synthesized speech in Table 6.4.

We can see how the WMR improves, which by informal expert listening, corresponds to fixing the previous attention errors of the Tacotron 2-like architecture. While the attention failures can make Tacotron 2-like synthesis generate an overall poor sample, FastPitch fails in a different way. The inherent noise in the data affects FastPitch giving two types of common failure: (1) samples cut at the end: as FastPitch makes use of forced alignment for the ground-truth duration during training, if the training data was not accurately cut when segmenting the utterances, the synthesized utterances might repeat this behaviour; and (2) intrusive non-verbal sounds: as the training data has sometimes interjections from the other speaker (back-channels) or short laughs, these are also replicated randomly by the model during inference. In the next section we propose a method to mitigate these errors.

6.5.1 Metadata vector

Using the metadata collected during the pre-processing stage described in Section 6.2 we create a metadata vector for every utterance in the training data. Each vector has three continuous values, all of them for the complete utterance: (1) percentage of speaker overlap, (2) speaker label confidence, (3) WMR. Recall that when selecting a subset of the data for training we already filtered using these values (see Section 6.3). During training, the vector is projected with a linear layer and summed to the encoder inputs of the FastPitch. At inference time, we synthesize with the "cleanest"

values: (1) 0% speaker overlap, (2) highest speaker label confidence level, and (3) 0% WMR.

We trained a model with the metadata vector and compared it to the Fast Pitch baseline. Table 6.5 shows WMR comparison over synthesized speech (300 samples). In Table 6.6 we compare FastPitch frame duration prediction error with respect to the forced alignment ground-truth (300 samples).

We can see that the use of the metadata vector improves the metrics in Table 6.5 and 6.6, which was confirmed through informal listening: the number of utterances cut at the end is reduced to a minimum. We also heard improvements in the presence of intrusive non-verbal sounds, as shown in an example utterance in Figure 6.7. Considering these improvements, all systems in further experiments made use of the metadata vector.

	WMR (%)	
System	**Speaker 1**	**Speaker 2**
Baseline	7.67	8.7
Metadata vector	7.46	8.08

Table 6.5: FastPitch baseline vs. conditioned to metadata vector, WMR comparison over synthesized speech.

	Frame error	
System	**Speaker 1**	**Speaker 2**
Baseline	0.502	0.345
Metadata vector	0.202	0.101

Table 6.6: FastPitch baseline vs. conditioned to metadata vector, frame duration prediction error comparison over synthesized speech.

6.6 Local coherence automatic analysis

Given the results in Section 6.4 we decided to take a completely different approach for the second set of experiments trying to answer the second research question. This time, we consider how all the methods that propose to model and condition text from context, including our method proposed in Chapter 5, rely on BERT embeddings. However, BERT embeddings are not transparent and it is not clear what are we encoding from the context through BERT that is useful for TTS. That is why in this section's experiments we use a method proposed by Jeon and Strube (2020), which combines concepts from linguistic Centering Theory (B. J. Grosz, Joshi, & Weinstein, 1995) for local context with large pre-trained language model representations, obtaining interpretable features derived from context. In the subsequent two sections, we will briefly describe the fundamental concepts from Centering Theory and the model proposed by Jeon and Strube (2020). Centering Theory has been applied in NLP to pronoun resolution, coherence assessment, automatic summarization, etc. Next, we will describe experiments to combine these approaches with TTS model training.

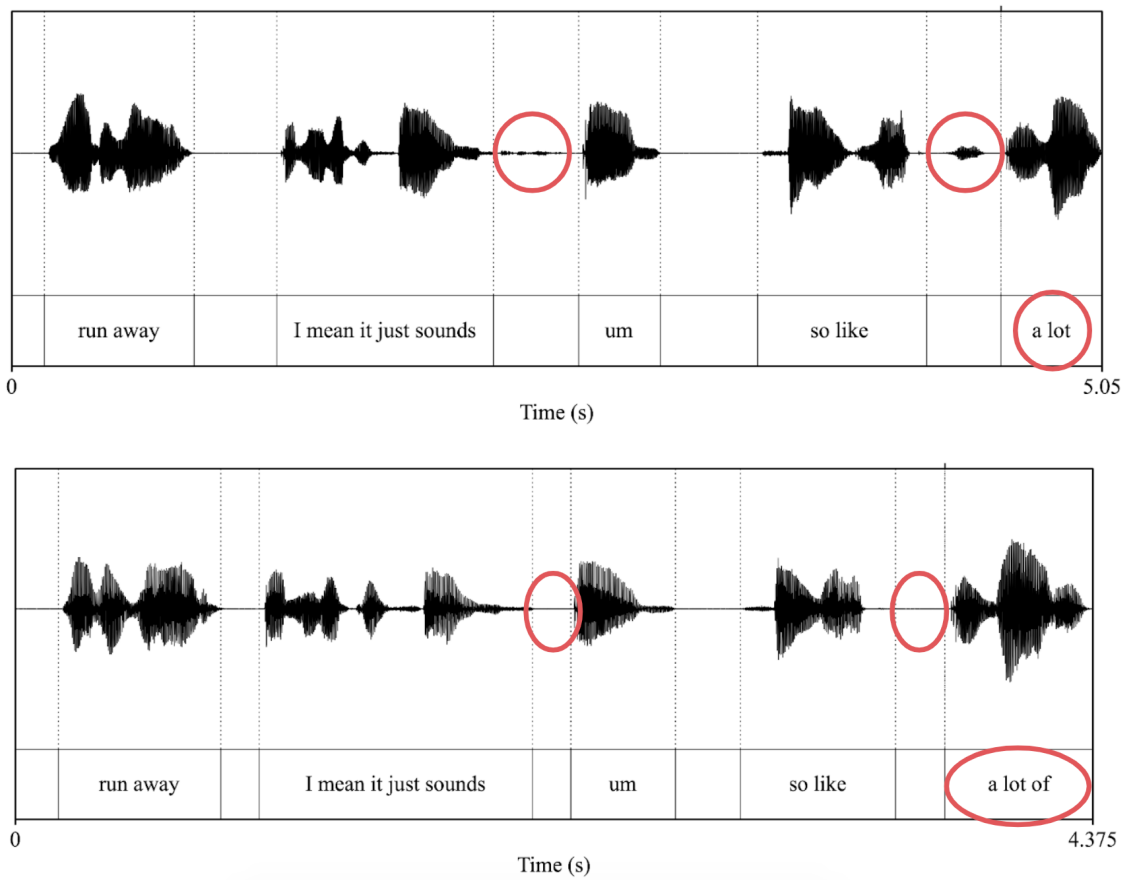

Figure 6.7: Comparison of intrusive non-verbal sounds with FastPitch baseline (top) and conditioning the baseline to the metadata vector (bottom). We can see that while the baseline shows two intrusive sounds in between the words of the utterance (circled in the waveform), the proposed model synthesizes these correctly as silences. Moreover, with respect to the end of the sentence, the bottom sample is no longer cut off at the end, unlike the top one, where the last word ("of") is missing.

6.6.1 Centering Theory

The Centering Theory (B. J. Grosz et al., 1995) describes local coherence and salience (focus) in a discourse structure. It has two fundamental concepts: center[5] and transition. A **center** is an element of an utterance, and **transitions** are relationships between centers of two consecutive utterances (e.g. utterance N and utterance N-1). According to the theory, a center can be of two types: (1) **Backward looking center** (Cb): an entity that links utterance N to utterance N-1; and (2) **Forward looking center** (Cf): an entity that is a candidate to be linked to the backward looking center of the next utterance. Every utterance in a discourse has a **single** backward looking center (except for the first sentence) and a ranked set of **K** forward looking centers. The highest ranked forward looking center is simultaneously the **Preferred forward looking center** (Cp).

There are three types of transitions that can be established between centers. If

[5]We keep the US spelling for this word and for the name of the theory considering them technical terms.

the backward looking centers of utterance N and utterance N-1 correspond (semantically) to the same entity, there are two possible transitions: (1) **Continue**, if the preferred forward looking center of utterance N corresponds to the backward looking center of utterance N; and (2) **Retain**, if the backward looking center in utterance N is not the preferred forward looking center. This means that in the first case, the topic of the utterances is the same, while in the second, there is a smooth change in the focus of the utterances. Finally, if the Backward looking centers of utterance N and utterance N-1 do not correspond to the same entity, there is a (3) **Shift**, e.g. the topic has changed more abruptly. Tables 6.7, 6.8 and 6.9 show an example based on B. J. Grosz et al. (1995) to exemplify these categories.

The theory proposes that the coherence of a text is established through the local relationship between forward and backward centers between two consecutive utterances, where a text that is more coherent would contain more Continue and Retain transitions than Shifts.

Utterance	**Cb**	**Cf (ranked)**	**Transition**
John went to his favourite music store to buy a piano	-	John, store, piano	-
He had frequented the store for many years.	John	John, store, years	Continue
He was happy that he could finally buy a piano.	John	John, piano	Continue

Table 6.7: Example of Continue transition (B. J. Grosz et al., 1995).

Utterance	**Cb**	**Cf (ranked)**	**Transition**
John went to his favourite music store to buy a piano	-	John, store, piano	-
He had frequented the store for many years.	John	John, store, years	Continue
It was closing just as John arrived.	John	store, John	Retain

Table 6.8: Example of Retain transition (B. J. Grosz et al., 1995).

Utterance	**Cb**	**Cf (ranked)**	**Transition**
John went to his favourite music store to buy a piano	-	John, store, piano	-
He had frequented the store for many years.	John	John, store, years	Continue
Mary always visited that store.	store	Mary, store	Shift

Table 6.9: Example of Shift transition (B. J. Grosz et al., 1995).

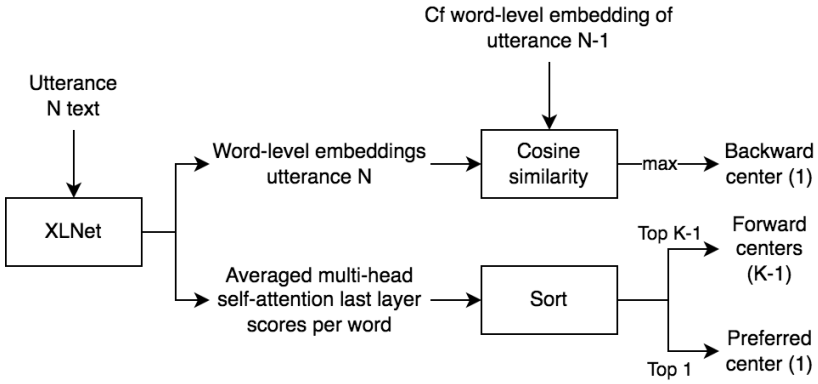

Figure 6.8: Centering analysis algorithm by Jeon and Strube (2020).

6.6.2 Automatic analysis

When analysing discourse to determine the centers and transitions using this theory, a linguist uses syntactic, pragmatic and discourse analyses. Jeon and Strube (2020) propose to approximate such an analysis given Centering Theory by automatically finding centers and transitions through an algorithm. Although their proposal has two parts, an algorithm and then a model, the algorithm is what produces simple and interpretable features, which are the ones we would like to use for TTS. Therefore, here we will focus on the algorithmic part of their proposal only. We made use of the original implementation by the authors[6].

The algorithm takes as input attention scores and embeddings at the word-level from a large pre-trained language model, from which it identifies centers and transitions. First, a document is segmented into utterances using Stanza (Qi, Zhang, Zhang, Bolton, & Manning, 2020). A document can be for example, a news article. Each utterance is passed through XLNet (Yang et al., 2019) to obtain word-level embeddings and the scores of the last attention layer of the model. Only the diagonal of the attention is extracted. XLNet is a popular large pre-trained language model that was used instead of BERT, as the authors obtained better results.

Centers are identified for every utterance, one at a the time. To find the forward looking centers, the diagonal self-attention scores of the last layer of the large pre-trained model are sorted. Special tokens and punctuation are ignored. The top K tokens are regarded as the forward looking centers, where the top one is the preferred forward looking center. To identify the backward looking center, they measure the cosine distance of each word-level embedding for the forward centers of the previous utterance, with a sentence-level embedding of the current utterance. The most similar is considered the backward looking center. Figure 6.8 shows a diagram of the procedure.

Transitions are identified considering two consecutive utterances at the time. First, following the theory described in the previous section, the algorithm determines by cosine distance of the embeddings, if the backward centers of utterance N and utterance N-1 are semantically close. If the distance is lower than a threshold, a Shift transition is assigned. Else, the algorithm measures the cosine distance between the embedding of the preferred forward center and the backward center. If the distance is low, a Continue transition is assigned, otherwise, a Retain one.

[6]https://github.com/sdeva14/emnlp20-centering-neural-hds

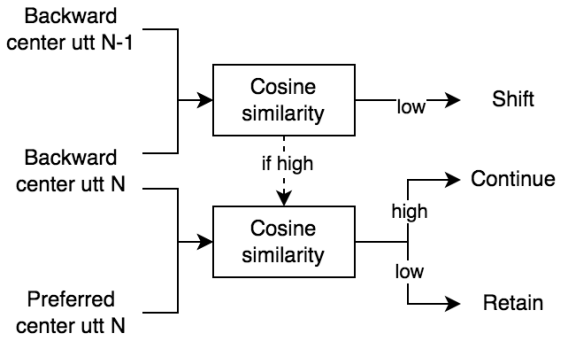

Figure 6.9: Transition analysis algorithm by Jeon and Strube (2020).

Figure 6.9 shows a diagram of the procedure. Notice that the algorithm is not trainable and its results solely depend on the self-attention scores obtained from a large pre-trained model and the thresholds chosen for the algorithm. Although we experimented with other thresholds, the ones defined by the authors in the original implementation seemed to work the best when applying the algorithm to a subset of 10 fragments from the podcast data.

6.6.3 Preliminary analysis

We now consider how to apply the automatic center and transitions analysis as features to enhance training of a TTS model with the podcast data. These features are encoding transparent and interpretable local relationships between utterances, as we know what the center and transitions categories mean. Table 6.10 shows a short fragment of the podcast data that has been run through the automatic algorithm to obtain center and transition analysis (K=3). We can see that although the analysis is not perfect, it still broadly captures the changes in topic and the relevant entities in the utterance sequence ("I, episode, him/he").

Before training directly with these features for the podcast data, we ran a preliminary analysis, to see if there was any prosodic relationship to the center and transition features when automatically obtained for the podcast data. As we would later train the TTS model with this data, instead of using Stanza for sentence segmentation, we used the utterances already segmented during the data processing described in Section 6.2. While center categories can be assigned as word-level features to a TTS input, the transitions can be taken as utterance-level ones (labels).

We extracted f0 and duration metrics for all the podcast data, both per utterance and aligned to the XLNet tokens to analyse correlations between the center and transition analysis. We analysed 300 samples per category, as well as the same words when not identified as a center, and group the analysis per speaker. See Tables 6.11, 6.12, 6.13 and 6.14. For the centering analysis, we analyse f0 and duration (in seconds) at the word-level. For the transition analysis, we analyse f0 and speech rate (in words per second) at the utterance-level. We measure statistical significant differences using a t-test.

Regarding Speaker 1 (Tables 6.11 and 6.12), some centers and transitions were significantly different in terms of duration/speech rate only. For the centers, all pairs compared were significantly different except for preferred vs forward centers. For the transitions, the only comparison not statistically different was Shift vs Retain.

Utterance	**Cb**	**Cf (ranked)**	**Transition**
One of our episodes this week was on Arturo Alfonso Schomburg.	-	was, our, week	-
Who as I said in the episode,	I	episode, said	Shift
Someone I'm embarrassed that I only became curious about in this very recent moment.	someone	curious, embarrassed, recent	Shift
Having just found a random mention of his name in another article,	found	random, mention, another	Continue
I find him fascinating.	I	him, find, fascinating	Retain
I'm enormously grateful for all of the work he did to put his collection together	I	collection, all, he	Continue
he was not at all the only person who was collecting books	he	collecting, was, book	Continue

Table 6.10: Example of automatic analysis applied to the podcast data.

Center type	**Mean f0 (Hz)**	**Std f0 (Hz)**	**Mean dur (s)**	**Std dur (s)**
Backward	214.8	46.61	0.27	0.21
Preferred	215.35	45.65	0.34	0.21
Forward	215.8	45.99	0.34	0.21
None	214.69	42.6	0.46	0.22

Table 6.11: Speaker 1 acoustic analysis of centers, at the word-level.

There was no statistical differences in f0 for either centers or transitions.

Regarding Speaker 2 (Tables 6.13 and 6.14), the center analysis shows that, all pairs are significantly different both in terms of f0 and duration, except preferred vs forward centers. In terms of transitions, only Retain vs Continue was significantly different, and only regarding speech rate. Transitions did not show significant differences in terms of f0.

We can see that the results are somewhat mixed. It is interesting that the two speakers show such a different behaviour (considering that they are both part of the same conversation): Speaker 1 showing differences in duration/speech rate only for both centers and transitions, while for Speaker 2 centers are more significant that transitions in overall. Knowing that not all comparisons are significant regarding these suprasegmental variables, we should not expect that including centers and transitions as features might have a large effect on the TTS results, but nevertheless these features could bring some significant improvements for the synthetic speech, helping with duration prediction for Speaker 1 and word-level prosody for Speaker 2. Finally, we notice that preferred and forward centers were never significantly different, which makes sense as theoretically these categories are only different in

Transition type	**Mean f0 (Hz)**	**Std f0 (Hz)**	**Mean w/s**	**Std w/s**
Shift	213.79	25.93	3.66	1.22
Retain	214.93	27.67	3.65	1.16
Continue	211.55	29.01	3.94	1.31

Table 6.12: Speaker 1 acoustic analysis of transitions, at the utterance level. Speech rate is words per second.

Center type	**Mean f0 (Hz)**	**Std f0 (Hz)**	**Mean dur (s)**	**Std dur (s)**
Backward	220.26	48.74	0.22	0.17
Preferred	212.57	51.03	0.33	0.2
Forward	212.73	50.98	0.32	0.2
None	201.58	42.71	0.44	0.21

Table 6.13: Speaker 2 acoustic analysis of centers, at the word level.

Transition type	**Mean f0 (Hz)**	**Std f0 (Hz)**	**Mean w/s**	**Std w/s**
Shift	204.18	29.23	4.01	1.27
Retain	205.54	29.19	3.91	1.27
Continue	205.05	29.12	4.12	1.34

Table 6.14: Speaker 2 acoustic analysis of transitions, at the utterance level. Speech rate is words per second.

terms of ranking (the preferred center is the most likely entity to be the backward looking center among the forward centers).

6.6.4 Text-to-Speech systems and listening test

As mentioned before, we propose to make use of the automatic analysis of centers and transitions for the podcast data as features to augment the TTS input. We believe that these features are capturing local contextual information derived from text, as based on Centering Theory. Given the results in Section 6.5.1, all the models are trained using FastPitch and including the metadata vector conditioning. As shown in the previous section, the two speakers seemed to have different characteristics regarding the automatic analysis, so we decided to train speaker-dependent models for this experiment. We used as training data only utterances with a context utterance, in order to have both transition and centers for all of them. Every utterance is labelled with the transition given. The labels are embedded through a trainable look-up table and summed to the encoder inputs of FastPitch. To encode the center analysis we tested two different possibilities:

- Encoding 1: build a vector of the same length of the phonemes by upsampling the word-level center analysis. Embed it, and sum to encoder inputs. We consider four categories: (1) word is a backward center, (2) word is a forward or preferred center, (3) word is not a center, (4) not a word (punctuation and other symbols).

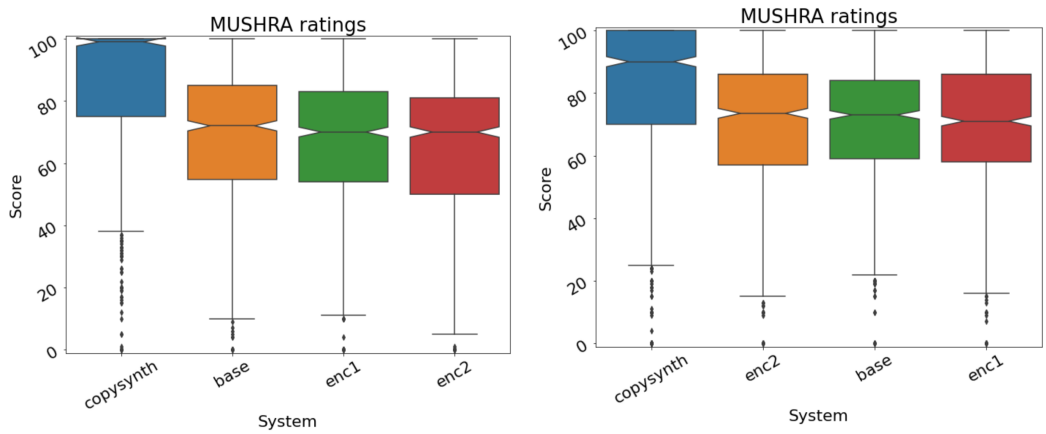

Figure 6.10: MUSHRA-like listening test results, for Speaker 1 (right) and Speaker 2 (left), comparing two ways to encode the center features.

- Encoding 2: include the center analysis as part of the input sequence, as token boundaries around the word that corresponds to the category. For example: "who as <Cb> I <Cb> <Cf> said <Cf> in the ...", and consider them input symbols in the same way as phonemes. For these symbols, the corresponding duration vector (needed for FastPitch's duration predictor), is assigned with zeros.

Therefore, we train and evaluate three systems:

1. Baseline (**base**)
2. Center Encoding 1 + Utterance encoding (**enc1**)
3. Center Encoding 2 + Utterance encoding (**enc2**)

We ran a MUSHRA-like listening test to evaluate the three systems enumerated above, including the copy synthesis of the ground-truth samples. We use 60 target utterances, 30 per speaker, and when presenting the samples to the participants, the text of both the target and the context utterance were shown (including a separation token between the two). 28 participants from an internal team of linguists took the test, and the test was implemented using an internal platform. Although participants are advice to give a 100 to one of the systems for each comparison, the listening test platform does not enforce participants to give a 100 to a certain utterance.

When selecting the samples for the test, to avoid ground-truth samples that were evidently not synthetic speech, we discarded any with laughter or other non-verbal speech that the TTS does not reproduce. We also measured the average fundamental frequency and duration of all the samples for all the systems, and included in the test those that were most different among each other, as a way of trying to see larger differences among the systems. From the remaining samples, 60 were randomly chosen. Figure 6.10 show the resulting scores.

For Speaker 1, Encoding 2 was significantly lower than the baseline, according to t-test. For Speaker 2, none of the compared systems was significantly different, although Encoding 2 had the best average score. On average, the copy synthesis score's for Speaker 1 were higher than for Speaker 2, although it was not 100 for the last one.

6.6.5 Discussion

Although the features we experimented with did not significantly improve over the baseline, it is interesting that the speaker differences match the preliminary acoustic analysis presented in Section 6.6.3: Speaker 2, for whom we saw differences in both pitch and duration related to the center categories, had a better score for Encoding 2, where the centers are explicitly encoded together with the input symbols. As we considered after the acoustic analysis, the amount of prosodic behaviour that we could link with these features was not necessarily going to help to explain all prosodic variation in the data. We consider multiple factors that explain the weak connections between the features and the acoustic behaviour. The Centering Theory was developed to explain written text, and one of the basic units is the utterance, which means that utterance segmentation plays an important role. Our data is spoken conversation, therefore utterance segmentation is not as precise as for text, possibly making the Centering Theory not optimal to analyse this data.

Unfortunately, we do not have gold standard labels of Centering Theory analysis for the podcast data, and they would be very costly to obtain through linguistic experts (which is why the automatic analysis is desirable), and therefore we do not know if the accuracy of algorithm was high, although as mentioned before, our inspection of some random samples showed that in overall the analysis was appropriate. It must be considered that given the characteristics of the data, some noise could have interfered in the automatic analysis, considering, for example, incorrect automatic transcriptions from the STT, that could lead to an erroneous analysis of the text.

In terms of modelling with the TTS system, the center features were sparse: an utterance would have at the most 4 words labelled with 2 different categories of centers. This would possibly make it hard to model very long utterances. This might explain why Encoding 2 worked better than Encoding 1, possibly because it improved in terms of the sparsity of the features otherwise encoded in a vector (Encoding 1), where most of the values would be not a center.

Finally, a minor detail to consider is that when we ran the listening test, one of the participants mentioned actually knowing the Podcast and being a regular listener. We do not know if having knowledge of the speakers could interfere in the judgements of the participants, or if other participants knew the podcast.

6.7 Conclusions

The experiments presented in this chapter had two main goals of interest for us and the Amazon team: test if we could use found podcast data from two-party conversational shows to train TTS systems, and explore further the use of contextual methods or features to improve these models. We managed to train models of reasonable quality with the collected data. We contributed further evidence that podcasts are a suitable source to extract data for TTS, by using new shows that have not been used to train TTS before. At the same time, we observed the many challenges of working with conversational found data.

Regarding our research questions proposed in Section 6.1.2, we managed to build stable and intelligible baselines with the podcast data. First, the improvements came from using data augmentation and training together with studio data to pro-

vide clean samples for the Tacotron 2-like architecture to learn the implicit alignment through the attention mechanism. Next, we saw considerable improvements through Fast Pitch with explicit duration modeling and the proposed metadata vector. The improvements given by the metadata vector are encouraging when dealing with found data, leveraging all the efforts that go into the proposed pre-processing pipeline during TTS training.

With respect to research question 2, unfortunately we did not find significant improvements for the two experiments we proposed. Undeniably, the challenging characteristics of the data are an important factor explaining the results, but not the only one, as we described for each of the experiments. In the case of using context for VAE reference selection, we believe that further experiments would be needed comparing data sets of different size and characteristics, considering that the candidate samples should be exhaustive enough to cover the acoustic space of the speaking style. Otherwise, instead of selecting references, the approach should be to predict the z vectors from the latent space as proposed by Strelec et al. (2021), where context could be part of the method.

Features derived from automatic local coherence analysis based on Centering Theory, and used to augment the TTS input, did not significantly improved the synthetic speech naturalness. However, we saw in the preliminary analysis how they had a relationship, although weak, to prosodic characteristics. This is consistent with previous work that identifies prosodic correlates with local and global discourse structure (B. Grosz & Hirschberg, 1992; Farrús et al., 2016; Aubin et al., 2019). We saw interesting differences between the two analysed speakers: for one of the speakers the centering analysis seemed to correlate better with acoustics than for the other. It is possible that the style of the speaker was closer to a textual style, or that utterance segmentation better matched sentence units, yielding a better automatic analysis of centers and transitions. In the next section we describe how we might consider continuing exploring the concept of local coherence into our experiments.

Finally, the time scope of the internship was limited and we had to prioritize the experiments described above. There were many aspects of the data and other possibilities that we could not explore, most notably the two-party aspect. Future experiments using two-party conversational data and local context should take into account the interaction dynamics between the speakers. For example, we noticed how each podcast show would oscillate between monologue phases (where one of the speakers might be telling a story or giving an opinion, without extensive interruptions of the other speaker) and dialogue phases with increasing turn taking between the speakers. We did not focus on this aspect and we will return to working with monologue data in subsequent chapters.

6.8 Chapter contribution

As explained in the introduction of this chapter, all the work described was carried out during the process of an internship in Amazon. This is why the chapter branches off with respect with the others, especially considering the use of dialogue data instead of monologue data. Considering the difficulties of dealing with the dialogue found data, we will revert to working with monologue data only.

Despite the negative results from the experiments leveraging context, there were two relevant conclusions that will guide our future experiments in the next chapters:

(1) The interesting differences between speakers, which will make us consider how our methods proposed in previous chapters could interact with data characteristics such as speaker identity or speaking style, and how this has not been systematically investigated, and (2) the correlation of local coherence features with acoustic characteristics, which we consider encouraging to look further into these theories, considering frameworks for local coherence more oriented to spoken discourse. The next two chapters will work systematically with these two ideas. Moreover, we will be able to reuse the data processing pipeline presented in this chapter in the following one, to prepare additional podcast data, now for single speakers.

Part III

Analysis and evaluation

Chapter 7

Context and data characteristics

7.1 Introduction and motivations

At this point we would like to remind the reader of the outline of the experimental work of this thesis, given in the Methodology chapter. In Part I, which includes Chapters 4 and 5, we proposed a method to condition a TTS system on a local linguistic representation, for which we found significantly better naturalness when compared to the baselines, evaluating isolated sentences. In Part II, corresponding to Chapter 6, we explored alternatives approaches to incorporate the local linguistic context, without finding significant improvements. Part I and II provided evidence to answer RQ1 from the Introduction, regarding the implementation of a method to leverage local linguistic context for TTS models. Considering the results, for the remainder of this thesis (which constitutes Part III), instead of exploring further methods, we discard the alternatives that did not provide significant improvements and focus only on the method proposed in Part I, specifically, our best system from Chapter 5 which was called **DS-utt + BERT-word**. Recall that this was the method that combined the use of Deep Spectrum to represent the acoustic context at the utterance level and BERT to represent the text context at the word (token) level. From now on we will refer to it as *our method to condition a TTS system on context*. Unless stated otherwise, this is the method we use for all systems conditioned on context. In the next chapter, which is the last experimental chapter, we will evaluate our method in context, looking to answer RQ3 from the Introduction. In this chapter we will investigate the limitations of our method regarding various data characteristics, focusing on speaker identity and style (but also considering others), looking to answer RQ2 from the Introduction.

We have three different motivations for this chapter. First, we have so far tested our method using only one data set, LJ Speech. This was a reasonable starting point for our experiments as it is widely used, and known to provide good results for TTS. Moreover, it was used by other work in parallel to ours on the use of context for TTS (G. Xu et al., 2020). However, we believe that data with a more spontaneous style could provide a source with stronger patterns at the local linguistic context level (see Section 1.2). Considering this, we looked for data sets with a more spontaneous style that we could use instead of LJ Speech. However, all the spontaneous-style data available to us is found data that includes other challenges (similarly to those for the collected data in Chapter 6). Therefore, we decided that we could use these found data sets to explore limitations of our method. Although we know that,

considering the challenging nature of the data, we will not be able to obtain high quality synthesis with it, we can still use it for in-depth analysis of the workings of our proposed method.

As a second motivation, in Chapter 6, where we experimented with two speakers from found podcast data, we saw that even speakers who came from the same conversational data set led to different results for the methods proposed in that chapter. This suggests that improvements provided by the use of local linguistic context might depend on speaker identity and speech style. This was further motivation therefore to explore other data sets, providing a multiplicity of speakers to test our method with and compare results.

As a final motivation, although we found positive results in Chapters 4 and 5, we believe that it is methodologically correct to test the generalization power of our method: e.g. would we get the same sort of improvements if applied to different data sets with different characteristics? Or, considering an alternative perspective, is our method robust to challenging data sets with diverse characteristics? We believe that these are important questions to consider. Even if we find that our method has limitations, this will not undermine the results already reported in previous chapters, but increases our understanding of what the method is doing.

Crucially, from this chapter onward we carry out all subjective evaluations for synthetic speech synthesized through our method using **synthetic acoustic context**. When we presented the results of Chapter 5 at the 2021 Speech Synthesis Workshop, we received feedback regarding the ecological validity of using ground-truth acoustic context to generate speech with our method. In a real-world use case, such as audiobook generation, only synthetic acoustic context would be available. Would then our method provide the same results when using synthetic acoustic context, through the mechanism described in Section 5.4.2? This was a question that we had already considered necessary to research in the aforementioned section, and therefore, we decided to move away from ground-truth acoustic context so as to study whether there is any variation in the found improvements when synthesizing with synthetic acoustic context.

The main goal of this chapter is to find the limitations of our best method so far, in order to providing further understanding on its workings. We propose to do this by training and testing our method to condition a TTS system on context on different challenging data sets with diverse, but relevant, characteristics. We investigate four different characteristics: speaker variability, data size, recording condition consistency and utterance segmentation length. We selected these variables based on our own intuitions gathered while performing all the experiments reported so far in this thesis. Additionally, we provide support from the literature, which we review in Section 7.1.1. For each of these variables, we postulate a hypothesis in Section 7.1.2. Each one will be considered in a subsequent section, finishing with conclusions (Section 7.6) and the contribution of this chapter (Section 7.7).

7.1.1 Related work

As explained above, we consider four data set characteristics for the experiments in this chapter. Because the relative effect that these characteristics might have can depend on the architecture of the TTS system, we focus on papers that study these variables for state-of-the-art sequence-to-sequence architectures only (for example,

HMM-GMM-based TTS models require very little data compared to sequence-to-sequence based models).

Speaker variability: dissimilar quality of synthetic speech when using the exact same TTS architecture for multiple speakers is a common finding (Lőrincz, Stan, & Giurgiu, 2021; Williams et al., 2020). In this chapter we refer to this phenomenon as speaker variability. Despite the empirical evidence on speaker variability in the TTS field and experimental evidence that certain phonetic phenomena are speaker-dependent (Section 1.2), little research has aimed to explain why and how speaker identity interacts with TTS architectures.

In our search for literature in the topic, we found only a couple of papers (Lőrincz et al., 2021; Williams et al., 2020) where speaker variability is researched for state-of-the-art TTS architectures, regarding training and evaluation. Lőrincz et al. (2021) trained a multi-speaker Tacotron 2 baseline for 37 speakers of Romanian from a parallel data set, e.g. in which each speaker utters the same text, but in their own style. The data set consists of approximately 50 minutes per speaker. They evaluate the synthetic speech for every speaker using two objective measures: WER and Equal Error Rate (EER, a speaker similarity metric). Among all speakers, the results vary widely: the best WER was ∼8% for one speaker, and the worst was ∼36% for another one. While WER could be attributed to good/bad diction, similar differences are seen for EER: for some speakers it was as low as 0% while for others, as high as 25%. Unfortunately, they do not perform listening tests to evaluate the synthetic speech, arguing it was not feasible given the large number of comparisons.

Focusing on evaluation, in a paper where I collaborated on the experimental design and provided some of the TTS systems, Williams et al. (2020) tested the use of deep learning models to automatically score synthetic speech quality, achieving a high correlation with MOS scores. The available MOS scores were obtained when evaluating synthetic data from many TTS and voice conversion architectures, all trained with the same large set of speakers. The results showed that specific speakers were consistently rated better or worse regardless of the architecture and the task. However, we do not know which characteristics are the ones that yield such differences.

We believe that the lack of research regarding this topic is mostly due to the inherent costs: a great number of speakers need to be collected, many models trained and evaluated. Even in papers where multi-speaker architectures are proposed, it is not reported if the listening tests were performed by testing one or several speakers from the training data (see, for example the Deep Voice 3 paper by Ping et al. (2018)). Therefore, although we will not solve or explain the speaker variability problem in this thesis because it is out of scope, we do consider it fundamental to test the robustness and generalization capabilities of our proposed method to condition a TTS system on context with respect to speaker identity.

Data size: in general, for every machine learning task, including TTS, we assume that with a larger data set, better results will be achieved. However, the improvements obtained by adding more training do not increase linearly with the amount. Chung et al. (2019) explored the minimum amount of data necessary for a stable Tacotron 2 model, training with an internal US English speaker. They informally find that using below 3 hours of training data, the synthetic speech intelligibility

noticeably decreased, where near ∼20 minutes it is basically unintelligible. A data set from 3 to 10 hours yields a similar synthetic speech quality. Increasing the data size further improves the synthetic quality, but anything between 10 to 40 hours of data yields similar results.

Most TTS papers, when proposing new methods or architectures, do not explicitly test which are the upper and lower data size limits within which their reported results are consistent. A notable exception is, of course, papers building TTS systems for under-resourced languages, where data size is a fundamental variable. Otherwise, most researchers make use of the maximum amount of data available. We believe that testing our method to condition a TTS system on context with different data sizes will reveal any crucial limitation of our method. Recalling the research presented in Section 1.2, not all types of sentences necessarily present a strong local relationship to each other: this depends on the semantic and pragmatic content, which means that it is possible that pairs of sentences with a strong local relationship are sparse in any given data set. Therefore, our method might require a relatively higher amount of data than our baselines to ensure that utterances with such relationships are present and frequent enough to make it possible to learn the local context relationships.

Recording condition consistency: resorting to found data is in many cases necessary to find data sets with specific characteristics or to access very large amounts. This is what we did in Chapter 6 to collect dialogue data from podcasts. However, as we explained in that chapter, one of the most challenging aspects of using found data relates to inconsistent recording conditions. This means that, even for the same speaker, audio quality changes throughout the data in terms of ambient noise, reverberation, changes of microphone, among other characteristics. Because state-of-the-art TTS is based on indiscriminately learning all the content present in the audio representation used as a target of the supervised learning, inconsistency in recording conditions is *also* modelled, affecting the resulting synthetic speech. Hu et al. (2019) confirm this by training systems using large amounts of low quality public data. They saw that even with large amounts of data, speaker-dependent models are challenging to learn with a Tacotron 2 architecture. When training a single speaker system with 19 hours of public data and comparing it to a system trained with a smaller studio data set (13 hours), the later yields significantly better MOS scores. The system is not robust to the noisy data, which is probably not only inconsistent in terms of recording conditions but also in other ways, such as transcription quality. Moreover, Lőrincz et al. (2021), already mentioned above, also studied the relationship between recording conditions and the quality of the synthetic speech output, finding a direct correlation. Synthetic speech generated with models trained with home recordings have worse WER and EER than for studio recordings.

If recording conditions are inconsistent within the training data and such variation is not supervised in any way when training the model, it will decrease the quality of the synthetic speech. In our experience, we can informally report that inconsistent recording conditions in the training data can have a different effect depending on the model's architecture. For FastPitch, we have seen that most of the time the synthesis will reproduce the recording condition that occurs most frequently in the training data, while from time to time specific utterances will be generated

with other recording conditions seen during training (possibly due to overfit of specific word combinations). This irregular behaviour during inference is hardly ever desired from a TTS system. Therefore, we will usually try to model explicitly these differences either by subsetting utterances with homogeneous recording quality or by labelling utterances with recording quality characteristics (as we proposed with the metadata vector in Section 6.5.1). Because our method to condition a TTS system on context includes the use of acoustic representations from the previous utterance, we think that testing our method with data that has inconsistent recording conditions may provide interesting insights into how the method would cope with such variation.

Utterance segmentation length: it is common in TTS to make use of data sets that already come prepared for training, including being segmented at the utterance-level, either because it was recorded as isolated utterances or by manually or automatically segmenting larger audio units (such as chapters or paragraphs). In other cases, such as when we use found data, it is very likely that we will have to perform such segmentation, just as we did during data pre-processing in Chapter 6. In general, however, utterance segmentation is not questioned and it has been rarely analysed how different utterance segmentation criteria might have an effect on results. Possibly the only aspect that has been considered in previous research is the length of the utterances, although likely from an implementation concern: the longer the utterances, the more memory will be required during training, possibly needing to reduce batch size, and therefore making training time longer.

Therefore, as reasoned by Lenglet et al. (2021), long utterance segmentation is inconvenient for TTS compared to shorter utterances. To explore the effects that utterance segmentation length might have, they apply different segmentations to audiobook data for French, then training Tacotron 2 systems. They compare two different segmentations: the original segmentation found with the data set, with utterance duration ranging from 1 to 20 s, and a median of 6.44 s, and a new segmentation that yields on average shorter utterances, with a median of 2.77 s, where most of the utterances range from 1 to 5 s only. Considering objective scores in spectral distortion for the synthesized speech, the model using the shorter utterances for training was better. In contrast, listening test results showed that listeners significantly preferred the synthetic speech of the models trained with the original, longer, segmentation. This exemplifies that, although it is convenient to train with shorter utterances, this might not yield the best synthetic speech. Furthermore, if the data set is very strongly skewed to a specific utterance length, the model will have difficulties to generalize to unseen or under-represented lengths of utterance at synthesis time.

We consider that for our method to condition a TTS system on context, utterance segmentation might be crucial. Changes in segmentation length not only will have an effect on the resulting current utterance duration, but also on the duration of the context utterance, and therefore, of the representations extracted for it.

7.1.2 Hypotheses and methodology

For each one of the data characteristics we reviewed in the previous section, we test the next hypotheses:

- **H1 - Speaker variability**: synthetic speech quality with our method to condition a TTS system on context will significantly vary with speaker identity.
- **H2 - Data size**: our method to condition a TTS system on context will yield higher quality synthetic speech when more data is available.
- **H3 - Recording condition consistency**: our method to condition a TTS system on acoustic context will be affected by the consistency of the recording conditions.
- **H4 - Utterance segmentation length**: our method to condition a TTS system on context will be affected by utterance segmentation length.

As the hypotheses we consider are of different nature, we do not test them all in the same way. For H1 and H2 we will perform listening tests, as we need to prove that there are differences in synthetic speech quality. Because for H3 and H4 we only need to test if there any differences in the output (not necessarily better speech, but just a different outcome given differences in the modelling), we test them only by analysis and comparing validation losses. Unlike in previous chapters, all the listening tests in this chapter will be run using synthetic acoustic context, instead of the ground-truth, as described in Section 5.4.2.

Although ideally we would have preferred to use only one data set that could allow us to test all these hypotheses, we do not have such data. Therefore, for each hypothesis we use a different data set, selected specifically to test the relevant data characteristic. Thus, for the next four sections of this chapter, we will test one of this hypothesis, proving details on the data set utilized, and the results for each one.

7.2 Speaker variability

H1: synthetic speech quality with our method to condition a TTS system on context will significantly vary with speaker identity.

7.2.1 Data and systems

To test this hypothesis, we selected 3 speakers from the Parallel Audiobook Corpus (Ribeiro, 2018). We selected this data set for this experiment because the parallel nature of the data makes it the most appropriate, in the same way as the data set used by Lőrincz et al. (2021), allowing us to control that the only variable that changes is speaker identity, while text materials are controlled.

We selected the speakers labelled “sde” (7.18 hrs), “ekl” (7.65 hrs) and “mth” (7.84 hrs) from the “Emma” book, which we selected for being the longest, and these three speakers (there are another two available for that book) had the most similar quantities of data. The data set provides text annotations and utterance segmentation for all speakers. We align it with Montreal Forced aligner to obtain phonetic transcriptions and alignments.

All the speakers are female and have a UK English accent. We subset 3500 training utterances and 400 for validation (from the first 6 chapters of the book), per speaker. Notice that although the phonetic transcription inputs are the same

for every speaker, when we align the data with Montreal Forced aligner we obtain silence tags that we also use as inputs, and these might differ per speaker.

We consider this data set to be ideal to test the speaker-dependency of our method, but we also consider it challenging because the amount of data per speaker is not very large, the speakers are amateur, and moreover, the speakers have a tendency to perform "voices" for the characters in the book. Considering this, we trained one baseline system and one conditioned on context according to our method; both were multi-speaker. Both systems were trained for 300 epochs.

7.2.2 Listening test and results

Considering that we are only comparing two models per speaker, we ran a forced preference listening test. The test included a vocoded ground-truth reference, and participants were instructed to score which of the two options is the **most similar** to the reference. Including the reference is a way to limit the listeners judgements on how many possible versions of the speech sample might be preferred while we do not evaluate in-context yet (Latorre et al., 2014), and it is consistent with our past MUSHRA-like tests that include references. We do not include a "no preference" option: if there is no preference the results will simply not be statistically significant.

All three speakers with 30 (identical) text sentences each, were randomly sampled. Every participant listened to all speakers. The samples were organized in blocks per speaker (e.g. the participant listened first to all the samples from speaker 1, then all the samples from speaker 2, etc.). The order in which the blocks were presented was randomized. We recruited participants native of UK English using Prolific. We discarded participants that took more than twice or less than the duration of all audio in the listening test to finish it, resulting in 25 reliable participant results.

It is important to notice that this is the first listening test where, when generating synthetic speech with our method to condition a TTS system on context we use **synthetic acoustic context**, using the method described in Section 5.4.2.

Results are shown in Figure 7.1. We use the Binomial test (p-value=0.05) for statistical significance comparing "baseline" and "context" for each speaker independently. For "sde' and "mth' the preference for the baseline was significantly higher. Although for "ekl" the preference towards our model was slightly more, this was not significant. We compare the binomial distributions from one speaker to the others using chi-square. "ekl" vs. "sde" and "ekl" vs. "mth" were significantly different, while the distributions of "sde" vs. "mth" were not significantly different.

These results support our hypothesis H1: applying the exact same method to data from three speakers leads to preference results that significantly vary. This can be taken as further evidence that, depending on the speaker, acoustic relationships between two consecutive sentences might differ, even if the textual content of the sentences is the same, as stated in Section 1.2, and as was seen in the Related work (7.1.1) of this chapter. Although these results support H1, they do not explain why or what characteristics in the speech or style of "ekl" are slightly easier to model for our proposed method, in comparison to the other two speakers. It is possible that "sde" and "mth" might have a higher tendency to make "character voices" which cannot be easily predicted from the local linguistic context, and could be, furthermore, harming the predictions of our context method when using synthetic

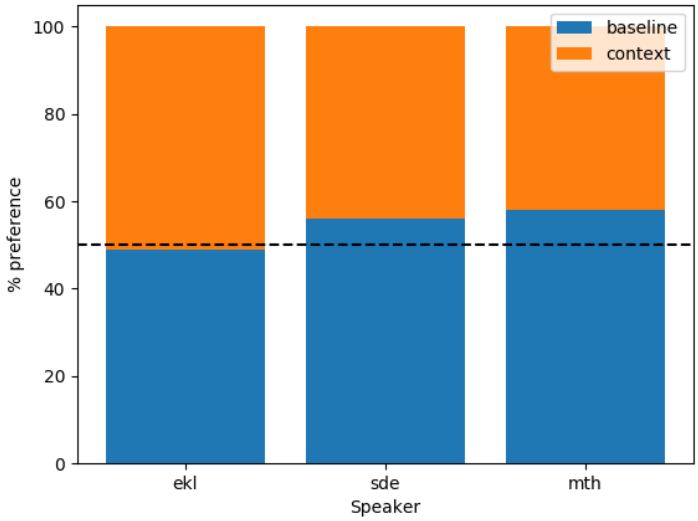

Figure 7.1: Preference test results for speakers from the Parallel Audiobook data set. "baseline" in blue corresponds to the baseline FastPitch architecture, while "context" in orange is our method to condition a TTS system on context. The dashed line marks the 50% preference.

acoustic context. However, we do not have an automatic way to measure the amount of "character voices" per speaker. From a more general perspective, answering this question might require a large scale listening test comparison of many parallel speakers which is out of the scope for this thesis. Nevertheless, in the next chapter we attempt an analysis that can bring some further understanding over speaker variability (see Section 8.3.3).

Finally, our proposed method did not perform successfully for this data set. We speculate that, together with speaker variability, the amount of data available might be relevant. It is possible that ∼7 hours of data per speaker are not enough to obtain significant improvements with our method. Unfortunately there is no further parallel data for these speakers in order to test this hypothesis. As such, in the next section we will attempt to explore the effect of data size using a different data set.

7.3 Data size

H2: our method to condition a TTS system on context will yield higher quality synthetic speech when more data is available.

In this section we experiment with how the amount of training data available might have an effect on the improvements seen when using our method to condition a TTS system on context. As the hypothesis states, we believe that, with more training data available, our method will perform better.

As we speculated at the end of the previous section, it is possible that the amount of training data that we had for each parallel speaker (about 7 hours) is too close to the *minimum* amount of data needed to make our context method work. Recall that, as described in the Related work in section 7.1.1 of this chapter, it has been

reported that below 10 hours of training data, the quality of the synthetic speech with a sequence-to-sequence architecture starts to decrease. If, additionally, we take into account that local linguistic context relationships might be sparse in the training data (as discussed in Section 1.2), our proposed model might need even more data than the baseline normally requires to obtain significantly better results. Finally, as we saw in the results of the previous experiment, we need to consider how speaker variability might interact with training data size. We collect a special data set to investigate all these topics which we describe in the next section.

7.3.1 Data and systems

To answer H2 we needed a data set as large as possible, including at the very least two speakers to provide speaker variability. Considering that there is no further data for the Parallel Audiobook speakers nor for LJ, we looked for a suitable subset of the Spotify Podcast data set (Clifton et al., 2020) which in total consists of 47k hours of transcribed audio, one of the largest data sets available to us. As in Chapter 6, a *show* corresponds to a specific podcast which comprises several *episodes*.

Although the data set provides text transcriptions (automatically generated through an STT API) and some available metadata, it is not known yet which podcast shows are suitable for TTS, and therefore we applied a procedure similar to the one described in Section 6.2 to find appropriate ones for our experiments. Considering the difficulties of working with dialogue data we saw in Chapter 6, and for the sake of consistency with the rest of the experiments, we aim to find single-speaker shows with several hours of data. This is by no means an easy task: inspecting the metadata file available with the data set, we identified 1893 shows tagged as US English (the largest group per language). It would be unfeasible for us to listen to them all and therefore we apply a semi-automatic process to find a suitable single-speaker subset.

First, we only considered shows that in total amount to more than 25 hours (which is similar the size of LJ), resulting in 129 shows. To have an automatic approximation of which shows might be single-speaker, we used the speaker tags that are available in the transcription files that come with the data set. Although these are not entirely reliable, they can be used as a starting point. We discarded any show that did not have at least 3 episodes that were not entirely assigned to one speaker tag only. That left 38 shows.

Considering that 38 shows is more manageable, we made a manual inspection to determine if they are single-speaker. We randomly listened to 2 minutes of three episodes per show, and discarded any show that evidently had multiple speakers (such as interview shows or multiple hosts). The remaining shows were only 6. Nevertheless, this random sampling for manual listening does not guarantee that **every** episode of the show is single-speaker, and it would be very costly to listen to them all. Therefore, next we made an episode-level analysis for those final 6 shows, to discard any episode suspected of having more than one speaker.

We started by segmenting every episode into utterances, and for every utterance we extracted a speaker embedding with the pyannotate library (Bredin et al., 2020). With the utterance-level speaker embeddings as input, we iteratively performed K-means clustering for each **show** separately, using K=2. Our assumption was that most of the utterances of a show will belong to one cluster only if the show is single

speaker, while the second cluster will be smaller, possibly capturing only irregular utterances. If, in contrast, there was more than one speaker, these two clusters will have more similar sizes.

Although we clustered utterances at the show level, we want to discard suspicious episodes. To do so, using the resulting clusters, we determined for each utterance the cluster it was assigned to. Then, we considered them per episode. If more than 70% of the utterances of the episode belonged the largest cluster (at the show level), then the utterances from that episode were kept, otherwise, discarded. With the new filtered set of utterances we performed clustering once again, and the same discarding procedure, until no utterances were discarded.

Shows that were most likely (and consistently throughout episodes) single-speaker remained with most of the data originally collected. Otherwise, shows that were suspected of either being completely multi-speaker or having many such episodes were left with very little data after our procedure. We selected at least two shows so that we could test if there any differences concerning speaker variability. Therefore, finally, we select the two shows left with the most data: we will call them "Show 1" (69.9 hrs) and "Show 2" (40 hrs). These can be considered speaker labels in the same way as in the previous section "ekl", "sde" and "mth".

We built 3 data sets for the two selected shows, considering the number of utterances, to make it easier to match to previous data sets. The data sets, each per speaker, had 3 sizes: Small (**S**), which consisted of 3.5k utterances, which was approximately equivalent to the data size of the Parallel Audiobook speakers in the previous section, allowing to test if this is the minimum required for out method to work. Medium (**M**), which had 12.4k utterances, similar to the number of utterances for LJ. And Large (**L**), which had 17k utterances, which was the maximum available considering the smallest of the two shows. For each data size we trained a baseline system ("Base") and a system with our method to condition on context ("Ctxt"), in both cases, multi-speaker: 6 systems in total. For every data set we used the same 600 validation sentences per speaker, extracted from held-out from episodes.

Although this data set was the best one available to test the hypothesis in this section, it is in many ways very different to our previous data sets (apart from the data used in Chapter 6). The most important differences are:

- Speaker gender: the two podcast speakers are male, while our previous speakers were female. Although this is not in itself more challenging, the male speakers have a very different style to the female ones: for example, presenting a considerably smaller pitch range.

- Style: the podcast speakers are relatively more spontaneous than our previous speakers, which were all reading audiobook data.

- Recording quality: it is very likely that not all episodes for the podcast speakers have the same recording quality. In contrast, our previous speakers had consistent recording qualities. While this is something we will investigate further in the next section, at the very least we should keep it mind here as, for example, the vocoder performs noticeably worse for the podcast speakers (we tried to fine-tune it to the podcast data but the resulting output was purely noise).

- Found data: the data will present some of the issues we saw regarding the use of found data in Chapter 6, including any errors that might resulting from the pre-processing of the data, such as utterances that might not be perfectly segmented or errors in the automatic transcription.

We should keep in mind these differences while analysing the results and extracting conclusions from the experiments, as they might not be directly comparable to, for example, the experiments with the Parallel Audiobook speakers. Nevertheless, we think we can still test H2.

Finally, although we could have leveraged the metadata approach we proposed in Section 6.5.1 when handling the conversational podcasts in that chapter, we could not apply the same approach for this data. In this case, we do not have access to a reliable second STT to obtain further word error rates with which to label the data.

7.3.2 Listening test and results

For this evaluation, we needed to test 6 systems, evaluating each speaker individually, and therefore we use a MUSHRA-like design 2.5.1. As in our previous MUSHRA-like tests, we included a reference, for the same reasons we gave for the previous experiment in Section 7.2. Because the number of systems we were comparing is already quite large (6), we did not include the reference hidden within the systems compared. To make it more similar to the previous experiment in this chapter, participants were instructed to rate the systems in terms of **similarity** to the reference, scoring from 0 to 100.

Samples were grouped in two speaker-dependent blocks. Sample selection was challenging: we randomly sampled test utterances and discarded the ones that contain domain specific vocabulary (video-games domain), until selecting 30 samples. As before, the acoustic context for the context-conditioned models was synthetic, following the procedure in Section 5.4.2 to synthesize the held-out episodes. Participants were screened to be native speakers of US English to match the podcast speakers accent. 25 participants were considered for the final results after filtering according to the time taken to complete the test[1] and discarding a couple of participants that suffered technical issues.

Results are shown in Figure 7.2 for each speaker. Significance tests were performed with a Wilcoxon signed rank test using Bonferroni correction to account for the number of comparisons. Tables 7.1 and 7.2 show the statistically significantly different pairs.

Overall, for none of the comparisons our method to condition a TTS system on context was significantly better than baseline. It is possible that, despite of the amount of data, our proposed context method is simply not robust to the pre-processing data noise in found data. Regardless, we can still make some interesting observations from these results (we will refer to these observations in later paragraphs by number):

[1]We do not know why but participants took particularly long to complete this test, therefore we had to extend the maximum amount of time from twice the duration of all audio in the test to three times as much, in order to keep the same amount of participants as for the experiment in Section 7.2.

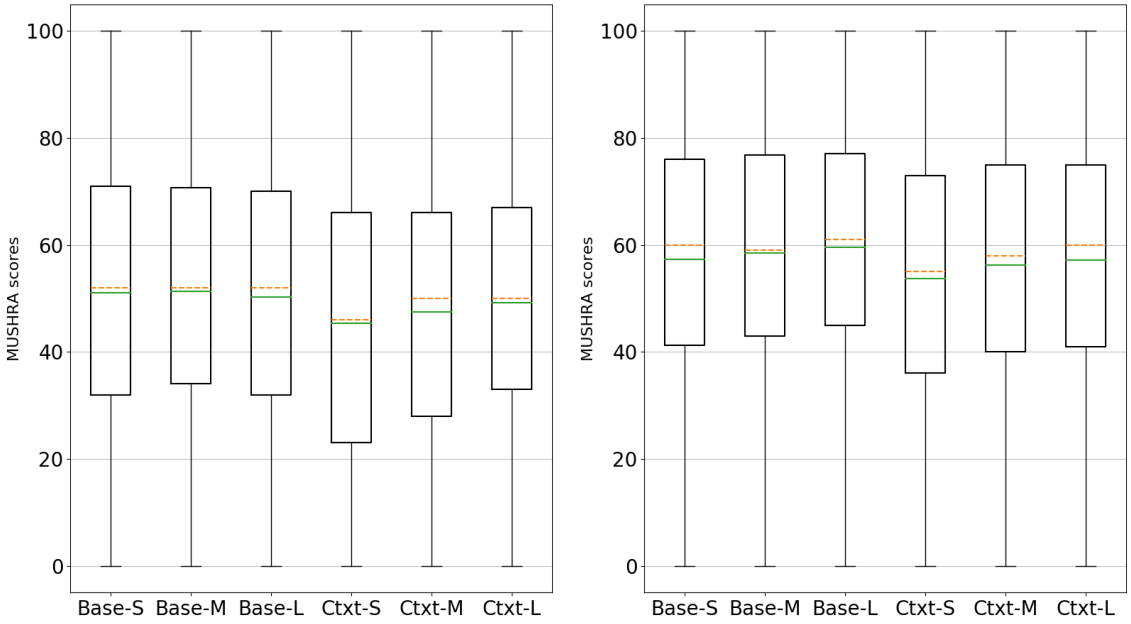

Figure 7.2: MUSHRA-like listening test results for Show 1 (left) and Show 2 (right). "Base" corresponds to the baseline architecture, "Ctxt" to our TTS model conditioned to context. **S**, **M** and **L** correspond to training data size, small, medium or large, respectively. Inside the boxes, solid lines correspond to the mean; dashed lines to the median.

Base-M					
Base-L		X			
Ctxt-S	X	X	X		
Ctxt-M	X	X	X	X	
Ctxt-L	X	X		X	X
	Base-S	**Base-M**	**Base-L**	**Ctxt-S**	**Ctxt-M**

Table 7.1: Show 1 significant comparisons.

1. *Speaker variability*: by simply observing Tables 7.1 and 7.2 we see again that the results differ per speaker, just as in the previous experiment. This will be considered for the rest of the observations in this list therefore.

2. *Baseline vs our method to condition a TTS system on context*: for Show 1, most of the scores for the baseline were significantly better than the scores of the systems using context (with the exception of **Base-L** vs **Ctxt-L**). In contrast, for Show 2 the baseline scores were significantly higher only for the scores from our method with the same data size or smaller.

3. *Within baseline comparison*: for Show 1, only **Base-M** scores were significantly better than **Base-L**. It seems as though the increased amount of data was harmful for the baseline rather than beneficial. In contrast, for Show 2 there is a significant difference between **S** and **L** only.

4. *Within context models comparison*: for Show 1, the scores for each system conditioned on context were significantly better than the scores for the system

Base-M					
Base-L	X				
Ctxt-S	X	X	X		
Ctxt-M		X	X	X	
Ctxt-L			X	X	
	Base-S	**Base-M**	**Base-L**	**Ctxt-S**	**Ctxt-M**

Table 7.2: Show 2 significant comparisons.

using a smaller training data set. Meanwhile, for Show 2 the systems conditioned on context **S** vs **L** and **S** vs **M** are significantly different, but **M** vs **L** is not.

Considering the results, we can reject **H2**: although with more data available, in most cases, our method to condition a TTS system on context will bring significantly higher improvements using a larger training data set than a smaller one, these improvements are not significantly better than the baselines. We see speaker variability irrespective of training data size.

If we compare the trends of the baseline systems with respect to the systems with our method, the baselines were mostly not significantly different from each other (only one comparison per speaker). In contrast, the models with our method to condition a TTS system on context exhibit a trend of improvements with each data increase for both speakers (with the exception of **Ctxt-M** vs **Ctxt-L** for Show 2). We can think that it is possible that if we had a model with an even larger data set these scores might keep increasing with our method to condition a TTS system on context; in contrast, the baseline, providing that maintains the same trend, would stay the same.

Moreover, the results raise further questions: first, it is possible that some speaker characteristics are better suited for our method to condition a TTS system on context while being irrelevant for the baseline architecture, e.g. increasing data improves our systems conditioned on context but not the baseline system (which we will further investigate this in Section 8.3.3); second, in noisy conditions (caused by pre-processing of found data), it is possible that our method to condition a TTS system on context requires very large amounts of data in order to bring significant improvements in comparison to the baseline. Unfortunately, we do not have further data to test the latter (considering both speakers), but in the next section we will consider a related issue, the effect of having acoustic noise present in the training data through inconsistent recording conditions.

7.4 Recording condition consistency

H3: our method to condition a TTS system on acoustic context will be affected by the consistency of the recording conditions.

While carrying out the two experiments above, we noticed a difference in the behaviour of the two data sets when comparing the validation loss between our method to condition a TTS system on context and the baseline. Figure 7.3 shows

the comparison for the Parallel Audiobook and the Spotify Podcast speakers (recall that for both data sets we trained multi-speaker models).

We are interested in the relative difference between the validation loss of the baseline and our system. For the Parallel Audiobook speakers (left), the baseline has a slightly higher loss, but barely different to our system, compared with the validation loss difference for the Spotify Podcasts speakers (right). We hypothesize that this difference is due to inconsistent recording conditions for at least one of the podcast speakers. As the speakers from the Parallel Audiobook data set have consistent recording conditions, the improvements that our system can bring are prosody-related only (we saw the same for the LJ speaker in previous chapters). This is expected, because FastPitch uses an MSE loss for both pitch and duration, the overall mean between the baseline and our system will not be drastically different: the differences are seen in terms of emphasis and prosodic patterns within that mean (as we showed in the analysis in Section 5.4.1). Considering the contrast, we hypothesize that for the Spotify Podcasts speakers the acoustic context is helping to model the otherwise unexplained variation in recording quality (which as we've mentioned before, can be common in found data), which will help predict the mel spectrogram, reducing the overall loss. Notice that the local minima in the first epochs for both models corresponds to the warm-up steps defined when training FastPitch.

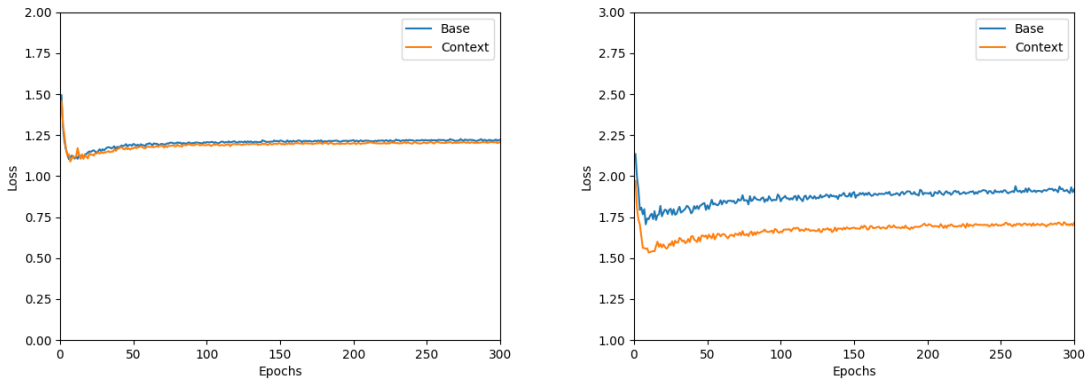

Figure 7.3: Validation loss for baseline (Base) and our method to condition a TTS system on context (Context), trained using data from the three speakers from the Parallel Audiobook from Section 7.2 (left) and from the two speakers of the Spotify Podcasts from Section 7.3 (right). Notice that the limits of the y-axis are not the same, but have the same range.

To test **H3**, in this section we trained a model with our method to condition a TTS system on context and a baseline system using training data with inconsistent recording conditions. Because we are not aware of the extent of these differences in the podcast data, we will utilize a data set for which we know the inconsistencies in recording quality from previous work (Dubiel, Halvey, Oplustil-Gallegos, & King, 2020). We present an analysis of the resulting generated speech to understand what the context is modelling for this data.

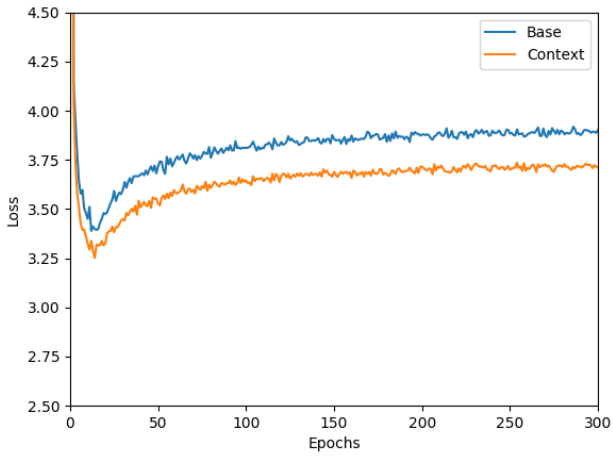

Figure 7.4: Validation loss for the baseline system (Base) and our method to condition a TTS system on context (Context), for the RG speaker of the IBM Debater data set using acoustic context only. The Context model has a lower validation loss than the baseline.

7.4.1 Data and systems

We trained a baseline system and our method to condition a TTS system on context for one of the speakers ("RG") of the IBM Debater corpus (Mirkin et al., 2018). The data set contains several monologues for many speakers, which each consists of a speech of ∼5 minutes where they argue in favour or against a controversial topic, in a semi-spontaneous style. We had previously worked with this data set and so we know that recording conditions vary greatly across different monologues. We selected this speaker as it has ∼24 hours of available data (similar to LJ). We use only one speaker and focus on the inconsistent recording conditions only.

We segmented the monologues and obtained 11938 training and 1119 validation utterances. Our method to condition a TTS system on context was trained only using acoustic context in order to focus on its effect, independently of textual context. Figure 7.4 shows the validation loss for the two systems. We can see that the difference between the baseline and the system trained using our method to condition a TTS system on context is closer to the difference for the Spotify Podcasts speakers than for the Parallel Audiobook speakers (comparing to Figure 7.3). This can be taken as evidence that at least one of the two Spotify Podcasts speakers, very likely, has inconsistent recording conditions similarly to the IBM speaker.

To provide further evidence for this hypothesis, we train two new systems, one baseline and one using our method to condition a TTS system on context, but this time including monologue-level labels. This means that every utterance is also conditioned on an embedding of a label specifying the monologue it belongs to. The embedding is summed to the encoder input, together with the acoustic context. As we said before, our inspection of the data showed that recording quality varied between monologues. By adding a label identifying the monologue, a good portion of the unexplained variation should be accounted for the baseline too, reducing the gap between the two validation losses. Figure 7.5 confirms this. We can see that although the baseline+labels loss is slightly higher than our model's, the difference

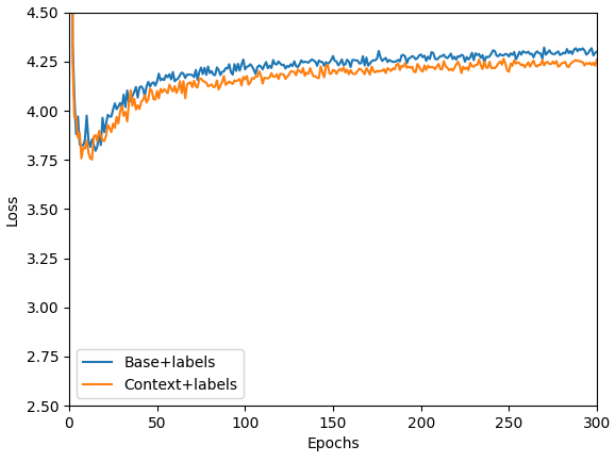

Figure 7.5: Validation loss for baseline (Base) and (Context), for the RG speaker of the IBM Debater data set using acoustic context only, this time, conditioning on utterance level monologue labels too.

between the two has been reduced, showing a pattern more similar to that for the speakers from the Parallel Audiobook data set. We believe that the loss absolute value is higher because the labels are not perfect: groups of monologues share similar recording conditions.

7.4.2 Analysis

We hypothesize that the acoustic context is capable to model in detail the recording conditions of the previous utterance, which will very likely be the same conditions of the current one, therefore, aiding mel spectrogram prediction. We illustrate this through a set of spectrogram plots shown in Figures 7.6 and 7.7.

At the top of each figure, there is the ground-truth spectrogram which will be used as acoustic context to synthesize with our method to condition a TTS system on context: Context A in Figure 7.6 and Context B in Figure 7.7. We can see how they differ in terms of frequencies: Context A presents less energy for more frequencies than B, which is perceived as muffled speech, while Context B has better quality. At the bottom of each Figure, we see the spectrogram corresponding to an identical text sentence, the Target utterance, where both versions had been synthesized using our method to condition a TTS system on context (without labels). In each figure the same Target text has been synthesizing using the context on top: in Figure 7.6 using Context A as acoustic context, and in Figure 7.7 using Context B as acoustic context. We can see how the Target presents similarities in frequencies to the Contexts as described above (the circles indicate locations of clear contrast).

Considering this analysis and the validation loss exhibited above, we can confirm **H3**. This provides further evidence to the conclusions we gave in Chapter 5 that the acoustic context is acting as an abstract unsupervised label. For this particular data set, where recording conditions are inconsistent, the acoustic context is encodes the recording conditions present in the context, and the target utterance is synthesized "replicating" such conditions. The encoding and transference of recording conditions is very accurate, as the Deep Spectrum combined with GST seems to capture the

frequencies of the acoustic context with great detail. Capturing this variation helps to reduce the validation loss.

This rises the question: at training time, or when synthesizing using ground-truth acoustic context, there is a clear source from which to "replicate" the recording conditions. But, what happens at synthesis time **without** ground-truth acoustic context?, e.g. when we initiate synthesis with our "initial" vector (as described in section 5.4.2)? To test this, we synthesized multiple samples with that initial acoustic context and inspected the spectrograms of the generated samples. We also synthesized the same samples with the baseline. It seems that both of them default to possibly the most common recording condition in the data set, which as described in Section 7.1.1, is the behaviour we see in FastPitch when training with inconsistent recording conditions. We cannot confirm that this corresponds to the most frequent recording condition in the data set but, this confirms that we can still synthesize successfully with our model without the ground-truth context for this data set.

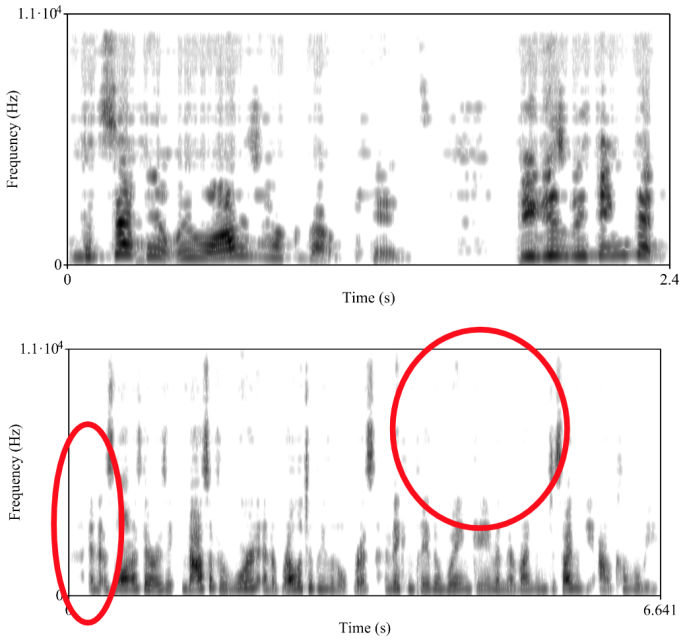

Figure 7.6: Spectrograms to illustrate the effect of acoustic context and recording conditions. Top spectrogram correspond to ground-truth Context A. Bottom one is the synthesized Target utterance, generated conditioning on Context A, using our method to condition a TTS system on context. The circles indicate how the frequency characteristics of the context utterance is replicated on the target utterance.

The fact that the acoustic context, when training our method to condition a TTS system on context for this particular type of data, is capturing and replicating the recording conditions when synthesizing a new utterance can be considered both advantageous and not ideal. Although it is a behaviour that might not be relevant for an application at first, we can think of a use case for audio editing, which is when a portion of a longer audio file needs to be edited. Some authors had proposed to use TTS for such task, showing how taking into account the context is crucial to generate a sample that successfully fits with the rest of the audio that does not need to be edited (Morrison et al., 2021). Our method would consistently produce

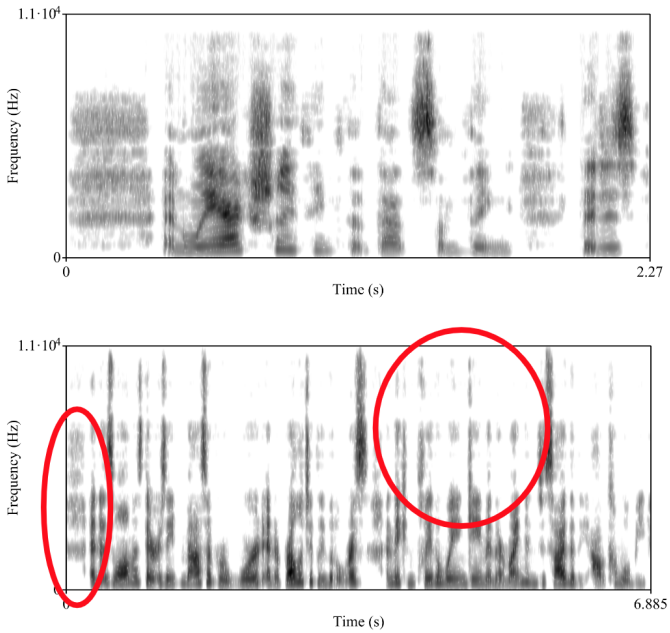

Figure 7.7: Spectrograms to illustrate the effect of acoustic context and recording conditions. Top spectrogram correspond to ground-truth Context B. Bottom one is the synthesized Target utterance, generated conditioning on Context B, using our method to condition a TTS system on context. The circles indicate how the frequency characteristics of the context utterance is replicated on the target utterance.

samples that also closely match the quality of the rest of the audio.

In a more general sense, we can see that for this type of data, the acoustic context is picking up on the most evident relationship between the utterances, possibly disregarding prosodic or semantic relationships, which is the part we are most interested in modelling. Therefore, if we want to avoid to model the recording conditions through the acoustic context, it would be possible to suppress that information from the acoustic representation, for example, using delexicalized acoustic context (as we showed in Section 5.4.3.1), or other tailored representation.

Finally, going back to confirm if the differences in validation loss seen for Spotify Podcasts speakers are due to inconsistent recording conditions, we re-trained the models independently and inspected the behaviour of the validation losses. We saw the gap between validation losses (similarly to Figure 7.3) only for Show 2. We confirmed this by informal listening of clips from different episodes, where we could notice differences in room acoustics and, sometimes, background music. This could have affected the linguistic improvements that our method could have brought, and could explain why there was not a significant improvement from **Ctxt-M** to **Ctxt-L** for Show 2, as the method would have been mainly encoding audio quality characteristics rather than making the speech more natural.

7.5 Utterance segmentation length

H4: our method to condition a TTS system on context will be affected by utterance segmentation length.

In this last experiment we test if improvements brought by our method to condition a TTS system on context can be affected by different ways of segmenting the training data into utterances. Considering that our context method is based on representations that are extracted from the context utterance, it is possible that the length of the utterances can have an impact on potential improvements. For example, Deep Spectrum features are a fixed-length vector that represents the acoustic context and are extracted for the complete utterance: the longer the utterances, the potentially greater information loss.

Although utterance segmentation can be based on different criteria, such as linguistic analysis to find grammatical sentences or other more sophisticated methods, we only consider here differences in length.

7.5.1 Data and systems

Although ideally we would test **H4** using one of the previous data sets, such as the Spotify Podcasts, to avoid possible confounds with inconsistent recording conditions or pre-processing derived noise, we made use of a fourth speaker from the Parallel Audiobook data set: "msm", which is the largest speaker available when combining all the different books. The 3 books sum up to 17.24 hours. We held-out 8 chapters as validation data (4, 1 and 3 chapters from each book, proportionate to their total length).

The Parallel Audiobook data set provides both a segmented version and the complete chapters for every book. Therefore, in this case, unlike for the first experiment in this chapter, we made use of the complete chapters. We aligned them using the Montreal Forced aligner (McAuliffe et al., 2017) to obtain timestamps and phonetic transcriptions. We segmented the data set three times to obtain three different versions in which on average utterances are either long, medium or short. Based on the word-level alignments, we grouped words into utterances and cut at the word boundary where the accumulated duration exceeds a certain threshold. We set three different thresholds to obtain the different data set versions: 3, 6 and 9 seconds. Figure 7.8 shows the histograms of the utterance lengths for the three data sets. We can see that although the total amount of hours is the same, the number of total utterances changes depending on the segmentation used. For each data set we trained a baseline and our method to condition a TTS system on context, making six models in total.

7.5.2 Analysis

Just as in Section 7.4, instead of using a listening test to confirm or disprove our hypothesis, in this section we analyse and compare the validation loss of the models trained (See Figure 7.9). Recall that the validation sets for each system had been segmented with the same criteria as for the training data, but they all share the same chapters as validation data. We can see that both the baseline and our method to condition a TTS system on context display the same pattern (although the differences are very small): the systems trained with the longest utterances have the highest loss, the ones trained with the medium-length ones are in between, and the systems trained with the shortest utterances present the lowest validation loss. Considering that the relative order of the systems is the same for both the baselines

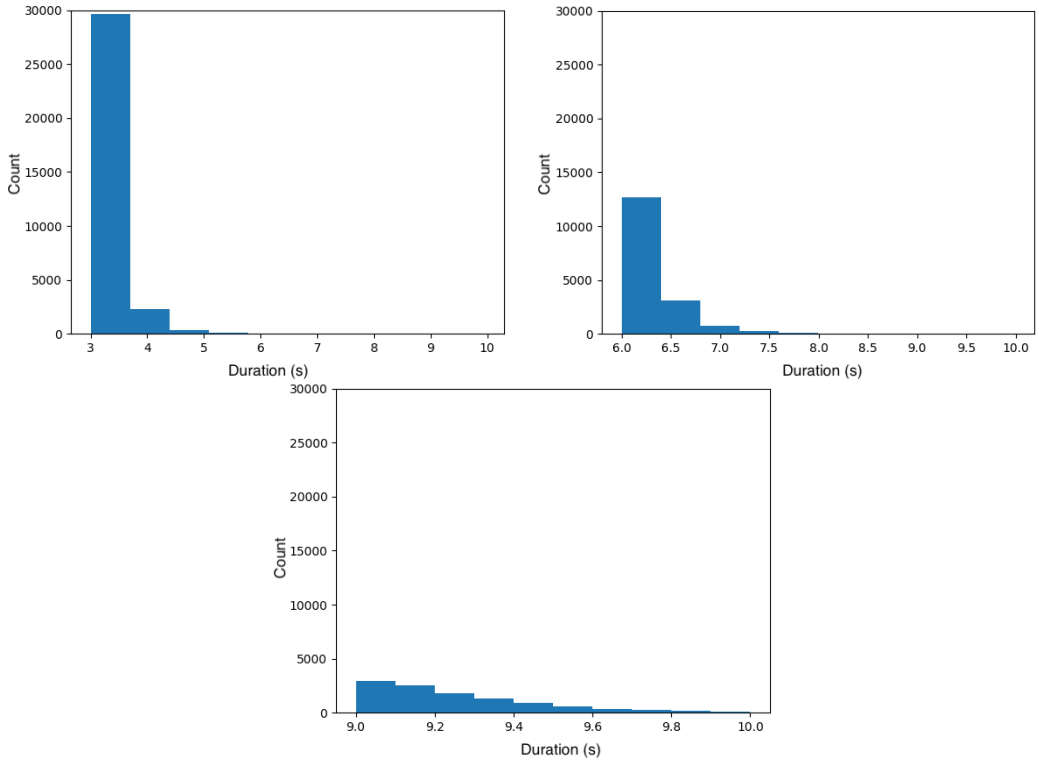

Figure 7.8: Histograms of utterance duration for the three obtained data sets through different utterance segmentation duration thresholds. Top-left corresponds to the shorter sentences, top-right to the mid-length ones and bottom, to the longest utterances. Notice that the range in seconds is different for each plot.

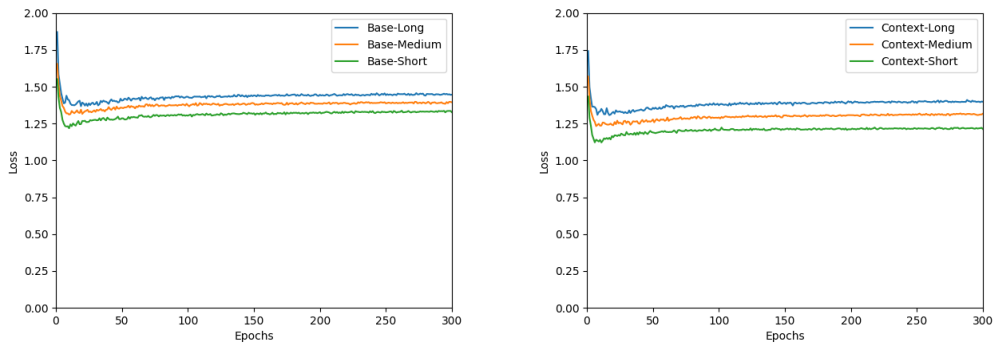

Figure 7.9: Validation loss for the models trained with "msm" data with three different utterance lengths: the baseline systems (left) and the TTS models conditioned to local linguistic context (right).

and the systems conditioned to local linguistic context, we can conclude that it is the main architecture, FastPitch, that benefits from shorter utterances.

This seems like a reasonable result if we consider that the MSE losses will be slightly better for shorter segments where less information is lost when averaging. This is expected behaviour and it is not particular to our method to condition a TTS system on context. Therefore, we can disprove the hypothesis given that while the length of the training utterances has an impact on the overall synthetic speech quality, it is related to the base architecture rather than the use of context.

7.6 Discussion and conclusions

In this chapter we proposed to investigate how different data set characteristics interact with the functioning of our context method which had produced significant improvements so far in this thesis. We considered four characteristics that had been identified as relevant both in the literature and from our empirical experience. These experiments allowed us to understand better how the method works and what its limitations are regarding certain data characteristics. The results will also allow us to answer **RQ2** from the Introduction chapter.

We applied the best method obtained in Chapter 5 to new data sets with different characteristics: speakers from the Parallel Audiobook data set, two speakers from the Spotify Podcasts data set and one speaker from the IBM Debater data set. These data sets were carefully chosen in order to test the hypotheses of Section 7.1.2. Moreover, it is good scientific practice to diversify the data used to experiment and develop new methods for TTS.

We evaluated our hypotheses with both listening tests and validation loss comparisons. For the first time in this thesis, the listening tests were run using synthesized speech from our method to condition a TTS system on context using synthetic acoustic context, making it closer to a real-world use case.

In Table 7.3 we summarise the hypotheses tested and the corresponding findings. We consider that speaker variability is a wider topic that needs to be investigated for TTS. Within the scope of this thesis, there seem to be some speakers whose speech style is better suited for our method than other speakers. This means that for some speakers is easier to predict the current sentence given the previous one than for others. This is consistent with previous work reported in Section 7.1.1. Whether this predictability is related to proficiency or speaker style, we do not yet know. We could devise an automatic way to select these speakers, which we explore in the next chapter. Regarding the other three data characteristics, we have provided enough evidence to show how our method is affected by them.

Finally, we discuss our findings in terms of robustness. Considering the data characteristics analyzed and the challenges present in the data, we can say that:

- Our method to condition a TTS system on context is **not** robust to speaker identity. A method to identify which speakers to train systems that could be improved given our method needs to be developed.

- Our method is **not** robust to utterance-level characteristics that cannot be predicted from the previous utterance, for example, in the case of "character voices", when using synthetic acoustic context.

- Our method is **not** robust to label noise, e.g. if the transcriptions are incorrect. This type of noise will certainly affect both types of context, acoustic and textual, making it harder to find the local linguistic relationships between the utterances.

- Our method is **not** robust to small amounts of data. Larger amounts of data are needed to learn the local linguistic relationships. The required amount of data probably depends on the other data set characteristics too, e.g. in the presence of label noise more data will be required.

Hypothesis	**Finding**
H1 - Synthetic speech quality with our method to condition a TTS system on context will significantly vary with speaker identity.	Yes, when comparing listening test results for parallel speakers, we see that the obtained scores distributions are significantly different. However, the scores obtained by our method are not necessarily significantly better in comparison to the baseline.
H2 - Our method to condition a TTS system on context will yield higher quality synthetic speech when more data is available.	Although we see that most the models conditioned to context through our method trained with larger data sets show significantly better listening test scores than those trained with smaller ones, the results are not significantly better than the baseline. Also, results depend on the speaker identity.
H3 - Our method to condition a TTS system on acoustic context will be affected by the consistency of the recording conditions.	Yes, we see that when there are inconsistent recording conditions these are encoded by the acoustic context. This can be beneficial if the task involves accurate recording condition generation, otherwise, it can be detrimental as the linguistic aspects of the context are no longer the main dimension modelled by our method.
H4 - Our method to condition a TTS system on context will be affected by utterance segmentation length.	No, differences in synthetic speech depending on utterance segmentation length seem to be related mainly to the architecture only and not to our method.

Table 7.3: Hypotheses regarding data characteristics and their interaction with our method to condition a TTS system on context and the corresponding finding after the experimental work presented this chapter.

- Our method **is**, in a way, robust to inconsistent recording conditions, as these are modelled by the acoustic context. However, linguistic relationships are obscured and recording conditions takes priority in terms of modelling for the method.
- Finally, all the items above interact with the fact that the synthetic speech quality seems to decrease further when using synthetic acoustic context in comparison to ground-truth acoustic context.

It is now clear why in previous chapters we found significantly better improvements in comparison to the baseline with the LJ data, but not with the speakers used in this chapter: LJ has consistent recording conditions, no acoustic or label noise, no "character voices" and a large amount of data. Moreover, when evaluating our method to condition a TTS system on context trained with the LJ data in

Chapters 4 and 5 we generated speech using ground-truth acoustic context. The effect of using synthetic acoustic context instead of ground-truth will be furthered explored in the next chapter.

7.7 Chapter contribution

This chapter is the first of the two that form Part III of this thesis. The aim of Part III is an in-depth analysis of the best method developed in Part I, in Chapter 5 and show its limitations and robustness to different data characteristics. In that sense, instead of testing further methods or further improvements to our method, a more useful contribution can be made by trying to understand what our method is doing and how is affected by data characteristics. That was the main contribution of this chapter. In this chapter we did not prioritize finding significantly better results but rather results that shed light on what our proposed method is modelling when we condition the TTS model on local linguistic context.

We address some of the remaining next steps previously established in Section 4.6 when prototyping our method: we started to consider systematically the data domain and speaker problem, by comparing the speakers from the Parallel Audiobook data set and including the Spotify speakers. These issues inspired **RQ2** from the Introduction chapter. We showed evidence of how some speakers are better suited for our method than others, and we superficially show some differences between read vs. spontaneous data. We ran listening tests using synthesized acoustic context for the first time, and we now suspect that some of the robustness issues seen for our method might get worse when not using ground-truth acoustic context. Both of these topics will be further investigated in the next chapter. Finally, we showed that chunking criteria are not as relevant as we thought for our method.

The experience we acquired handling the challenging conversational data in Chapter 6 was fundamental to prepare the data sets from this chapter. Furthermore, the effort to prepare the data sets used in this chapter will be leveraged for the next chapter too. The results found in this chapter were decisive for designing the final experiments in the next chapter, mainly in terms of further investigation of speaker variability, speaking style differences and the effect of using synthetic acoustic context. These matters are covered in the next chapter together with the application of in-context evaluation.

Chapter 8

Local coherence and in-context evaluation

8.1 Introduction and research questions

In this chapter we deal with in-context evaluation of synthetic speech. As the second chapter of Part III of this thesis, we continue looking for the limitations of the best method developed in Part I. While in the previous chapter we did this with diverse challenging data sets, in this chapter, we do it through evaluation, looking to answer the third research question from the Introduction to this thesis. This chapter presents two sets of experiments in Sections 8.2 and 8.3. As the final experimental chapter of this thesis, this is a logical way to end the experimental work: by taking an in-depth look at evaluation, as so far, we had only evaluated our methods through listening tests with isolated utterances.

When considering in-context evaluation we, once again, define context as the local linguistic context. When evaluating the current utterance, the previous utterance must be considered by participants, both in terms of text and speech, to provide a judgement that captures the relationship between the two utterances. This is in contrast to traditional evaluation where only isolated utterances are judged (reviewed in Section 2.5). It is also not to be confused with an "interactive" approach (Wagner et al., 2019; Mendelson & Aylett, 2017; Dubiel et al., 2020), which we will not research in this thesis.

In Section 8.2, we evaluate synthetic speech generated through our method to condition TTS systems on context with an in-context listening test. We make use of the listening test design proposed first by Clark et al. (2019) and later revisited in a collaboration with O'Mahony et al. (2021). We sought to answer two research questions through this listening test, both motivated by the fact that the TTS systems conditioned to our method are using the local linguistic context: **RQ8.1**: *Does speech generated by our method obtain significantly better listening test scores than the baseline's speech when evaluated in-context?* and **RQ8.2**: *Does speech generated by our method obtain significantly better listening test scores in-context than when evaluating isolated utterances?*

In Section 8.3, we propose to apply local coherence models from the NLP field as an objective in-context evaluation. In NLP, these models are trained to score pairs of sentences from a document. The model is presented with positive, coherent pairs, which are two sentences authentically sequential in the training data, and with

negative, non-coherent pairs, which are randomly sampled sentences, non-sequential in the training data. The model learns to give higher scores to the positive samples. Although these models are usually applied to text alone, some authors had already proposed methods to combine text with speech (Patil, Singla, Shah, Hama, & Zimmermann, 2020). We propose to use these models in two ways, with two corresponding research questions. **RQ8.3**: *Can we use a local coherence model to* ***analyse natural speech*** *in context and find relevant differences for speaker identity and speaking style?* Where natural speech is found to be more coherent, this can be taken as it having relationships between utterances at the local linguistic context level that should be easier to predict. **RQ8.4**: *Can we use local coherence models to automatically* ***evaluate synthetic speech****, as an objective in-context evaluation?* We consider that if our method to condition a TTS system on context is actually modelling the relationship between the two utterances, the generated current utterance will be more coherent when taken together with the context utterance, than for baseline generated speech, obtaining a higher score from the local coherence model.

8.2 In-context listening test

As Latorre et al. (2014), Clark et al. (2019) and O'Mahony et al. (2021) argue, in-context evaluation is always desirable, even when evaluating synthetic speech from models that do not take context into account. When evaluating utterances in isolation there are multiple prosodic versions of the same text that might be natural when considering one sentence only, but when we add the context, a smaller set of prosodic variability will be appropriate (Clark et al., 2019; O'Mahony et al., 2021). This should, for example, make agreement between listeners higher.

In the case of this thesis, we definitely need to evaluate our models in-context to see if our method can produce speech that is considered more natural in-context than a baseline that has not been trained with local linguistic context information. Furthermore, it is possible that our models produce synthetic speech that is more natural in-context than when considered in isolation. These two ideas are captured in **RQ8.1** and **RQ8.2**.

Although we consider that, given the nature of our methods, it follows that an evaluation in-context is a must, other authors researching the use of context to improve TTS do not provide a thorough discussion of the evaluation they carried out. For a review on this, see Section 9.2.2. We perform an in-context listening test in this section, giving a detailed account of the evaluation performed, and providing an in-depth discussion of the repercussions of both the design utilized and the results obtained. Although in the first experimental chapter we had an exploratory in-context listening test (Section 4.4.2), in contrast with the evaluation carried on in this chapter, we consider the listening test design more in-depth.

This section is organized as follows: in Section 8.2.1 we review in detail the work of Clark et al. (2019) and my collaboration with O'Mahony et al. (2021) which expands on Clark et al. In this collaboration I contributed with the experimental design in order to experiment with some aspects of the listening test design of Clark et al. that were ambiguous and that we needed to clarify in order to apply in this thesis. In Section 8.2.3 we apply as close as possible the listening test design from O'Mahony et al. (2021), using TTS models trained with LJ Speech. Finally, in Section 8.2.4 we discuss the importance of the results we obtained, how they can be

understood in relation to previous listening test results in this thesis and how they partially motivate the second set of experiments in this chapter.

8.2.1 Previous work and collaboration

Clark et al. (2019) proposed in-context listening tests to evaluate long-form synthetic speech (such as audiobook generation), arguing that the evaluation of isolated utterances is not optimal to measure whether prosody is appropriate for the context and if the speech is fluent in its long-form. Moreover, they hypothesized that evaluating utterances in isolation will not necessarily yield the same results as evaluating the same utterances in-context, given the interaction between the utterance and the context.

They compared isolated utterance evaluation with two in-context conditions: context+stimuli pairs (equivalent to our local linguistic context, where the stimulus is the current utterance) and a paragraph context. The stimuli sentences, i.e. the utterances that are being judged by the listeners, are the same across all conditions. They also included (non-vocoded)[1] ground-truth speech for all conditions. Additionally, they compared the use of textual context only, or the use of both textual and acoustic context. Finally, they also experimented with the use of ground-truth vs. synthesized context. Therefore, the conditions they tested and that we are interested in in this thesis were:

1. Isolated sentences: ground-truth vs. synthesized.
2. Paragraphs: ground-truth, synthesized.
3. Context + stimuli: ground-truth + ground-truth vs. ground-truth + synthesized vs. synthesized + synthesized vs. text + synthesized.

Importantly, the Wave-Net based TTS systems they trained to generate the synthesized samples are **not** in any way conditioned on contextual information, either during training or inference. They trained with a proprietary data set from the news-reading domain. Their listening test design was based on MOS, where they asked participants to score the stimuli from 1 to 5 in 0.5 intervals. The test was administered through a crowd-sourcing platform. The instructions participants received changed depending on the condition: if evaluating isolated utterances they were asked to score the **naturalness** of the samples; when in-context (either paragraphs or context-stimuli pairs) they were asked to evaluate the **appropriateness** of the stimulus. Each stimulus was rated 8 times. For in-context conditions, each utterance had a separate button to reproduce the context or the stimulus[1].

Although they experimented with both monologue and dialogue data, we will focus only on the results obtained for monologue. A result across all conditions was that the ground-truth speech always rated higher that the synthetic counterpart. In fact, the context and stimulus condition when both ground-truth, obtained the best scores. Most interestingly, when comparing the ground-truth scores over the three conditions they found significant differences. This already proved their hypothesis: listeners judgements **are** different when evaluating in-context vs isolated utterance evaluation.

[1]This was confirmed by the main author through private correspondence.

For the synthetic speech, conditions (1) isolated utterances and (2) paragraphs had the lowest MOS overall. This showed that stimuli in condition (3), when presented with the local linguistic context, were scored better than in isolation, even if the synthetic speech has not been generated with models using context in any way. They hypothesized that, for this condition, listeners take into account whether the overall quality of the stimuli is similar to that of the context, when asked to evaluate **appropriateness**. In contrast, it is more likely they will hear a mistake in the synthetic speech with a longer context, such as synthesized paragraphs, lowering those scores.

The main contribution of Clark et al. (2019) is that different listening test designs yield significantly different results for the exact same stimuli, something that, although intuitive it had not been previously systematically tested. Furthermore, their in-depth statistical analysis shows that there is no correlation between the scores between the three conditions: for example, the scores of the utterances rated in isolation cannot be used to predict the paragraph-level scores. This is very interesting and points to the fact that, as they conclude, listeners are truly judging differently among these three conditions and that in order to have a complete evaluation of the synthetic speech, both in-context and isolated evaluations should be performed.

We reviewed Clark et al. (2019) in detail because, although we found their results very relevant for our work, we wanted to explore further some aspects of the in-context listening test design. For our collaboration (O'Mahony et al., 2021) we focused only on the isolated vs context-stimuli condition. For a detailed description of the experiments performed and the results found please see the full paper. In here we focus on the aspects that are more relevant for the work in this thesis. We designed three new experiments:

1. Effect of instructions: considering that Clark et al.'s TTS system was not conditioned on context, could it be possible that the differences in scores, when comparing isolated utterances and context-stimuli pairs, was a confound effect resulting from using different instructions? To test this, we asked participants to rate naturalness for both isolated utterances and in-context, and, additionally, to rate appropriateness for utterances in-context.

2. Effect of between-sentence textual context dependency: as Clark et al. mentions, they did not pay special attention to the text materials being selected for the test. Could it be possible that context-stimuli pairs with a stronger linguistic dependency show differences in MOS scores? To test this, context-stimuli pairs with and without anaphoric relations were created.

3. Clark et al. hypothesized that participants are judging appropriateness of the prosody of the utterances when evaluated in-context. If this is correct, would manipulating the prosody of the stimulus into a prosodic pattern that is ill-fitting for the context (manipulated by the authors) result in a decrease in MOS score?

Only synthetic speech was used for all experiments (no ground-truth held-out speech). A DC-TTS-like architecture[2] was trained with the LJ Speech data set (Ito

[2] `https://github.com/oliverwatts/ophelia`

& Johnson, 2017), and speech waveforms were generated using Griffin-Lim. The methods developed by Suni et al. (2020) were used for manual manipulation of word prominence and prosodic boundaries required for the third experiment. Each experiment compared the evaluation of isolated utterances and context-stimuli pairs. The listening test materials were hand-crafted based on Wikipedia content. In total, 110 context-stimuli pairs were tested, using the same ones for all three experiments. As we had done previously in Section 4.4.2, the pairs were concatenated using a silence of 400 ms in between (in contrast to the use of two buttons in Clark et al.). The same MOS scale as in Clark et al. was used. The test was online and participants were crowd-sourced through Prolific. There were 104 participants for experiment 1, 144 for experiment 2 and for experiment 3.

The main findings per experiment were: (1) when asking for naturalness judgements for both isolated and in-context utterance evaluation, there are **no** significant differences between the MOS scores. When asking for appropriateness, the scores are higher in-context, replicating the results from Clark et al.; (2) the presence or absence of anaphoric text-dependency between context-stimuli pairs had **no** significant effect on MOS scores; and (3) manipulated prosodic patterns result in a significant effect on MOS scores.

From these findings, the first one is possibly the most important to us, showing that participants do understand something different when asked to rate naturalness vs. appropriateness. However, the results do not indicate which one should be used. However, we reason that if we are evaluating both conditions at the same time, isolated and in-context, it makes sense to ask participants to rate naturalness for both of them. In this way, if we obtain a difference in the results from the two conditions we will know that it is a consequence of differences in the speech and not an effect of giving different instructions per condition.

8.2.2 Synthetic context and exposure bias

Considering that all the data sets in the previous chapter had some challenging characteristic that might act as a confound, for this experiment we decided to revert to LJ, training all TTS systems with this speaker. As well as in Chapter 7, in this chapter we wanted to evaluate synthetic speech generated with our method using synthetic context. However, as we noticed in Section 7.6, there seems to be a decrease in the synthetic speech quality when using our method with synthetic context, in comparison to when using ground-truth context, for the speakers in that chapter. Although we briefly considered this for LJ in Section 5.4.2, upon closer inspection, when we synthesized the test sentences with our best method so far (**DS-utt + BERT-word**), we noticed the detriment in quality to be greater than we thought. The generated speech was noticeably more flat and there were more incorrect pronunciations than when using the ground-truth context. We confirmed our informal impressions with a preliminary listening test for isolated utterances with 10 participants only, where the system that used our best method so far with synthetic context was found worse than the baseline.

The mismatch between the use of the ground-truth context during training (i.e. teacher forcing) and the synthetic context during inference with our method is an example of *exposure bias*. Exposure bias was first identified as a problem by Bengio et al. (2015) for tasks associated with the use of Recurrent Neural Networks. Exposure

bias can diminish the quality of the model (or method)'s predictions, with errors that might accumulate over time for auto-regressive tasks. If a model or method has been trained with ground-truth information with certain characteristics that are not perfectly attained during inference, the model (or method) will not be able to perform at its highest quality during inference. This is what we are seeing in our method to condition TTS systems on acoustic context: at synthesis time the method depends on the quality of the speech generated by the system, which is certainly not on par with the ground-truth speech (otherwise, the TTS task would be solved).

In order to attempt to close the gap between the use of ground-truth and synthetic context, we decided to use a variation of our method (introduced in Section 5.4.3) to train the system for the in-context listening test. This version of the method is trained with delexicalized acoustic context. Additionally, during inference, fundamental frequency manipulation is applied to the acoustic context. By delexicalizing the speech we reduce the mismatch between the ground-truth context and the synthetic one in terms of potentially wrong pronunciations. By multiplying the synthetic context's fundamental frequency by a factor of 1.2 we can bring it closer to LJ's ground-truth speech. Informally, we saw that the synthetic speech generated with this variation of our method, although not as good as using the full method with ground-truth context, did produce speech that was more similar to LJ's natural speaking style. We perform the in-context listening test in the next section with this system.

Although we recognize that this is an aspect of our method that could be further improved, the goal of Part III of this thesis is no longer to improve the methods proposed in Part I and II. Rather, in this chapter our focus is on in-context evaluation, and therefore we limit ourselves to the variation of our method that can best bridge the exposure bias gap. Furthermore, following the in-depth perspective of Part III, in Section 8.3.4 we will use this problem as an opportunity to test an objective in-context evaluation that we will propose, by comparing the use of ground-truth and synthetic context. Lastly, in Section 9.3 we will briefly discuss the relative importance and potential improvements of exposure bias in our method and other methods that propose the use of acoustic context.

8.2.3 In-context listening test and results

To answer **RQ8.1** and **RQ8.2** we performed an in-context listening test. The work from here on-wards corresponds to novel work with respect to our collaboration with O'Mahony et al. (2021). We used the listening test design from O'Mahony et al. (2021) as closely as possible with just a few changes to adapt it to our needs: (1) we included held-out ground-truth vocoded speech; (2) instead of hand-crafted sentences we used held-out sentences from the training data as listening test samples. In this way we were able to include held-out ground-truth speech. Given that the text-dependency experiment in O'Mahony et al. (2021) did not result in significant differences, this choice of materials should not have been a problem; (3) we asked participants to rate **naturalness** both when rating isolated utterances and in-context; and (4) we used our method (with the variations detailed in the previous section) and a FastPitch baseline to generate the synthetic speech.

Notice that, unlike us, neither Clark et al. (2019) nor O'Mahony et al. (2021) used synthetic speech from models that are conditioned on context. We implemented

the test using the same MOS scale as described in the previous section. We semi-randomly sampled 71 stimuli from two held-out chapters from LJ (chapter 23 and 24), with their corresponding context utterances. We filtered any pairs of utterances that, when measured together, had a duration longer than 13 seconds.

We used the exact same instructions as in O'Mahony et al. (2021) for the two conditions:

- Isolated: "We want you to rate how **natural** the sentence sounds."
- In-context: "We want you to rate how **natural** the **second** sentence sounds given the *first* sentence."

For the in-context condition, the context-stimulus pair text was presented to the participants matching the font in the instructions: the first utterance in italics and the second in bold. The context utterance was presented with both text and audio, concatenating it to the stimulus with a 400 ms silence. For each condition therefore, we tested the ground-truth ("Vocoded"), the baseline ("Baseline") and our method (with the variations detailed in the previous section) to condition TTS on local linguistic context ("Context").

Every participant listened to both in-context and isolated conditions, each one as a block (within which presentation order was randomized), sampling a disjoint half of the sentences for each one. We recruited native speakers of US English, through Prolific, and administered the test with Qualtrics. As in previous tests, we discarded participants that took twice as much time to take the test compared to the total duration of the audio, resulting in 44 participants in total.

We had three hypotheses for this listening test:

- **H1**: synthetic speech generated with our method to condition TTS systems on local linguistic context will be rated significantly more natural in-context than the baseline.
- **H2**: synthetic speech generated with our method to condition TTS systems on local linguistic context will be rated significantly more natural in-context than when evaluating isolated utterances.
- **H3**: synthetic speech generated with our method to condition TTS systems on local linguistic context will not have significantly better scores than the baseline when evaluating isolated utterances.

The first two are related to the research questions. Additionally, we consider the third one as a possibility given that with this listening test design the participants will listen to both conditions and will be able to contrast them. It is also very likely that the scores for the system using our method might decrease with respect to previous listening tests in this thesis considering the discussion on exposure bias in Section 8.2.2.

Figure 8.1 shows the test results, on the left for isolated utterances and on the right for in-context evaluation. As the results were normally distributed, we used a two-sided t-test for significance, with Bonferroni correction. We compared both the results within a condition and across both conditions. For the isolated utterance

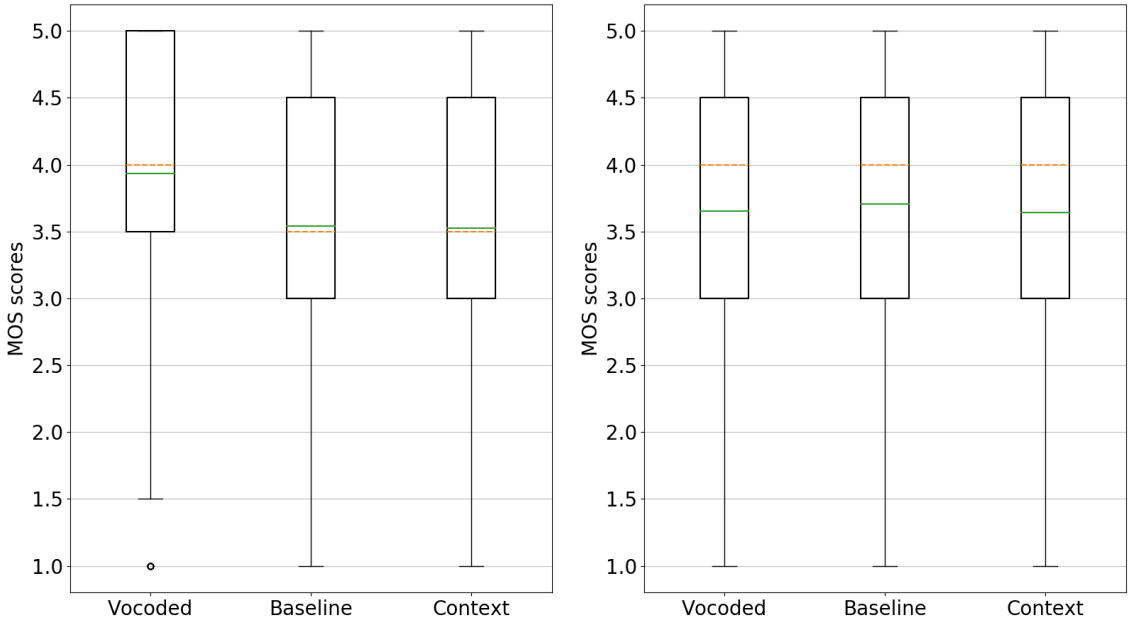

Figure 8.1: MOS scores for the listening test. At the left, for isolated utterances and, at the right, for in-context evaluation. Inside the boxes, solid lines correspond to the mean; dashed lines to the median.

condition, Vocoded was significantly better than both Baseline and Context, as expected. Baseline vs Context scores were **not** significantly different. Surprisingly, there were **no** significant differences between any of the systems in the in-context condition. Comparing across the two conditions, all comparisons are significantly different: in-context, both Baseline and Context systems have significantly better scores than the same system presented in isolation. Unexpectedly, the in-context Vocoded utterances were rated significantly **lower** than the Vocoded isolated utterances. Considering these results, we have to refute H1, as Baseline and Context results were no different in-context. We can confirm H2, although, for Baseline this was true too. We refute H3 because Context and Baseline results were not significantly different for isolated utterances. These results seem less important when considering the scores given to the Vocoded speech, which were quite unexpected, and so it seems very important to discuss these first.

8.2.4 Discussion

Vocoded speech results: our results showed that the Vocoded (ground-truth) speech in-context obtained significantly **lower** naturalness scores than the same utterances in isolation. This contradicts the results in Clark et al. (2019), where the condition of context-stimuli when both ground-truth speech, obtained the best scores. A couple differences might explain this disagreement: our data sets differ and their natural speech was not vocoded. In O'Mahony et al. (2021) there was no held-out ground-truth speech. The use of vocoded speech (rather than non-vocoded ground-truth speech) should not be a problem because all natural speech for the isolated utterances condition was also vocoded and results were as expected. Therefore, if we assume that the listening test design is correct, then we have to conclude that

listeners judge that LJ's ground-truth utterances are not as natural in-context as when considered in isolation. This is a very important conclusion, because if the local context relationships of the natural speech are very natural for listeners, even if our method manages to capture them, the listeners will not judge the synthetic speech generated through our method as natural when in-context.

Isolated utterances results: Vocoded speech received the expected score for isolated utterances. We must discuss though an apparent inconsistency in the results in this section with respect to listening test results in previous chapters. In Chapter 4, our best method using ground-truth context was significantly better than the baseline in a MUSHRA-like test for isolated utterances. Recall that in for the in-context listening test in this section, our Context system makes use of synthetic context with a variant of such method, described in Section 8.2.2, in order to bridge the gap between ground-truth and synthetic context. As we expected, the use of synthetic context, even through the method variation, was only as good as the Baseline for isolated utterances. This further supports our impression that the detriment of using the synthesized acoustic context is more relevant than we thought before.

In-context results: contrary to our expectations, there were no significantly different scores between Baseline and Context when evaluated in-context. By comparing Baseline vs Context only, we must conclude that our method is not able to learn the relationship between the utterances. However, bearing in mind the rest of the results, it could be that our method is learning the same "non-natural" in-context speech behaviour as seen for the ground-truth speech (with the additional detriment caused by the exposure bias). However, another possible interpretation of the results is that the in-context design is not capable of capturing the in-context differences, and that is why we did not see that the Vocoded condition was significantly better than any of the synthetic speech, as would be expected.

Isolated vs in-context results: O'Mahony et al. (2021) found no significant difference when evaluating isolated utterances vs. in-context if the instruction for the participants were the same, both rating the **naturalness** of the speech. We did not get this result with our evaluation, not only for the Vocoded speech, but actually for none of the systems, as all of them were significantly different when comparing across conditions. It might be possible that because participants did both the in-context and isolated conditions, they felt compelled to judge differently each (and maybe implicitly evaluating appropriateness when in-context, if they were exposed to the isolated condition first). Although this is possibly the least important of the results presented, it is an indication that more research is needed to fine-tune in-context listening test design.

From this discussion we highlight three questions that need to be addressed further: (1) Is LJ Speech less natural when heard in-context? Could it be possible that this is true for other speakers too?; (2) Is the use of synthetic acoustic context resulting in a significant decrease in the method's improvements than when using ground-truth acoustic context? (3) Is the in-context listening test design correct? We do not believe that we can fully answer the last of these questions as it is out of scope for our thesis. A systematic comparison of multiple combinations of listening

test design would require considerable resources. Instead, we take a different route for the final set of experiments in this thesis, in order to address these issues and some raised in the previous chapter, by proposing the use of a local coherence model from NLP that might be able to act as a proxy for an in-context evaluation.

8.3 Local coherence models

In the second set of experiments of this chapter, we try to address some of the issues raised in the conclusions from Chapter 7 and the conclusions from the first set of experiments of this chapter. Both sets of conclusions are closely related, and we believe we can address them both by using local coherence models.

As we explained in the introduction of this chapter, local coherence models score text documents, giving higher scores to more coherent ones. A discriminative approach is used to train these models, based on the use of positive and negative samples. Positive samples are pairs of sentences authentically sequential in the training documents, while negative ones are pairs randomly sampled. The assumption made is that a randomized document will lose the logic and semantic connections and the organization between ordered sequences of sentences, typical of a well-written coherent document (P. Xu et al., 2019; Lapata, Barzilay, et al., 2005). In Section 8.3.1 we explain in detail how these models work by introducing the fundamental papers on local coherence models.

Notice that although in Chapter 6 we made use of a type of local coherence model, that was different to the type of local coherence models we use in this chapter. The local coherence model in Chapter 6 was based on Centering Theory, and we used it to extract features to enhance the inputs for the TTS with word-level and utterance-level information. There are two fundamental differences with the experiments in Chapter 6 using local coherence models and the experiments in this chapter: (1) the local coherence models in this chapter are not based on a specific theory with limited concepts as the one in Chapter 6, but rather they have a more general definition of what coherence is. While in Chapter 6 the local coherence model used Centering Theory to design an algorithm, in this chapter, the local coherence models are trained on data; and (2) in this chapter we **do not** use the local coherence models to extract features for TTS (neither to train them together with a TTS system), we use them only to **analyse** natural speech and to **evaluate** synthetic speech.

Although local coherence models had traditionally been developed for tasks that involve text only, some authors (Patil et al., 2020) have proposed incorporating speech into these models. Patil et al. (2020) is reviewed in detail in Section 8.3.2 because we use their approach for our experiments in this chapter.

We believe that by taking into account speech too, we can identify pairs of utterances that have a stronger acoustic, more predictable, relationship at the local linguistic context level. In contrast, if the speech of a certain ground-truth "document" (which in the audio domain would be, for example, a chapter, or a podcast episode) has not been produced in a way that matches the meaning and sequential structure of the discourse, then it will be closer to randomized speech. We would expect to see this for both natural and synthetic speech, such that it will allow us to answer **RQ8.3** and **RQ8.4**.

In **RQ8.3** we want to see if by applying local coherence models to analyse natural speech in-context we can find significant differences per speaker and speaking style.

We saw in Chapter 7 the different results our method provided per speaker. Then, in the first half of this chapter, we saw how the natural speech for LJ was not considered more natural in-context. We hypothesize that the natural speech of some of the other speakers tested in Chapter 7 might share some similarities with LJ. This might be the reason why our method did not provide significantly better results in Chapter 7. Using the local coherence models to analyse the natural speech in-context should identify them. We include a speaking style comparison, as our data comprises either audiobook or spontaneous speech. We believe that the difference in the spontaneity of the speech might also be captured by a local coherence model. In Section 8.3.3 we train and test these models with the natural speech for most of the data sets used in Chapter 7, including LJ.

RQ8.4 will follow from the results given for **RQ8.3**. If a local coherence model, when applied to analyse natural speech gives consistent results, then we can test if we can apply the same model, trained with natural speech, as an objective evaluation of the synthetic speech in-context. As explained in the conclusions in Section 8.2, there are reasons to believe that the in-context listening test design requires further development, but this is not something we can do in this thesis. Instead, we propose to use the local coherence models as a proxy: an objective evaluation in-context.

While this would not be in any way a substitute for a listening test in-context, where listeners judge speech in a holistic way, it would be a measure to specifically evaluate if the local linguistic context relationship between the utterances generated through our method is stronger than the ones generated by a baseline. Such objective evaluation could be used for rapid testing and development of methods to improve TTS conditioning on context.

As an example, using the local coherence models to evaluate the synthetic speech in-context, we will further investigate the effect of using synthetic acoustic context (in contrast to using ground-truth one), an issue that was raised in Chapter 7 and in this chapter. This is a quicker and cheaper way of testing the effect of this variable, as otherwise, we would need to re-run all our speakers in an in-context listening test, using both ground-truth and synthetic acoustic context. We present this experimental work in Section 8.3.4.

8.3.1 Local coherence for written discourse

Coherence is a concept from Linguistics that is a property of text, and although many definitions can be found, we understand here that the coherence of a text depends on the semantic and logical connection of its sentences. A thorough discussion on the definition of this concept can be found in Y. Wang and Guo (2014). Automatic coherence scoring models are trained to predict a coherence score for a document. Ultimately, the goal is to obtain scores that correlate with human judgements for the coherence of a text (Lapata et al., 2005). These models have been applied to automatically evaluate second language proficiency or to evaluate automatic summaries, among other use cases.

Coherence is a complex concept, maybe similar to naturalness in TTS, since it encompasses multiple characteristics of a text. In this section we do not pretend to give an extensive review but rather explain how the local coherence model that we will use later in this chapter was developed. While some coherence models are based on a global analysis of textual documents, we focus on the models that work at the

local level. These models make the assumption that by scoring coherence locally, for pairs of sentences, and then averaging over all pairs in a document, the coherence of the document can be obtained. In this sense, this matches the definition of local linguistic context we have used throughout this thesis.

The first models that were developed for this task were based on finding entities in a document. The idea was to map out how the entities, and their relations, progress through a discourse, as a model of lexical cohesion (Lapata et al., 2005). The Centering theory reviewed in Chapter 6 belongs to this type of coherence model. These initials models had some disadvantages, such as relying on parsing and automatic lexical analysis and, given that they are mostly based on word-level relationships, lack power to capture more than a very limited aspect of what coherence is (J. Li & Jurafsky, 2016).

Discriminative methods became widely used in later models, starting with J. Li and Jurafsky (2016). These models, instead of trying to model a limited aspect of what coherence is, are based on a more general perspective, relying on data. Li and Jurafsky proposed a discriminative model trained with positive samples from authentic, correctly ordered, documents and negative samples from the same documents but randomizing the order of the sentences. Sentences are represented through the final timestep of an LSTM recurrent layer used to train a classifier. They obtain good results when testing the model to discriminate unseen positive from negative samples. One of the issues they encounter is that the model is not good at generalizing over unseen domains, and they hypothesize that negative sampling needs to be further developed to cover as much negative sampling space as possible.

Following the discriminative model proposed by Li and Jurafsky and focusing on improving some the mentioned issues, P. Xu et al. (2019) proposed a model for local coherence scoring with two main enhancements: sentence representations are obtained through large pre-trained models (GloVe embeddings) and the negative sampling method is refined. As in J. Li and Jurafsky (2016), positive samples are extracted from authentic ordered documents, and negative samples are only taken from the same corresponding document but randomized. Additionally, new negative samples are obtained at every epoch, and the same number of samples is extracted for every document regardless of the length.

Figure 8.2 shows the model's architecture. Embeddings of dimension d are extracted for every sentence in every document. Sampling is done of pairs of sentences, either with a negative or positive context. From the embeddings of each sentence in the pair a new representation that captures the distance between the two is calculated by concatenating: the two original embeddings $(s1, s2)$, the element-wise difference $(s1 - s2)$, the product $(s1 * s2)$, and the absolute value of the element-wise difference $abs(s1 - s2)$. The resulting representation is passed through drop-out and then fed to one 500-dimensional Linear feed-forward layer followed by ReLu and drop-out. The final output layer of dimension 1, returns the score for the sentence pair.

$$L = max(0, (margin - Scores_p + Scores_n)) \tag{8.1}$$

Finally, they use a margin ranking loss shown in Equation 8.1. $Scores_p$ are the scores given to the positive samples and $Scores_n$, the scores given the negative ones. The use of a margin and the max function ensures that the parameters of the network will only be updated if the difference between the scores is smaller than

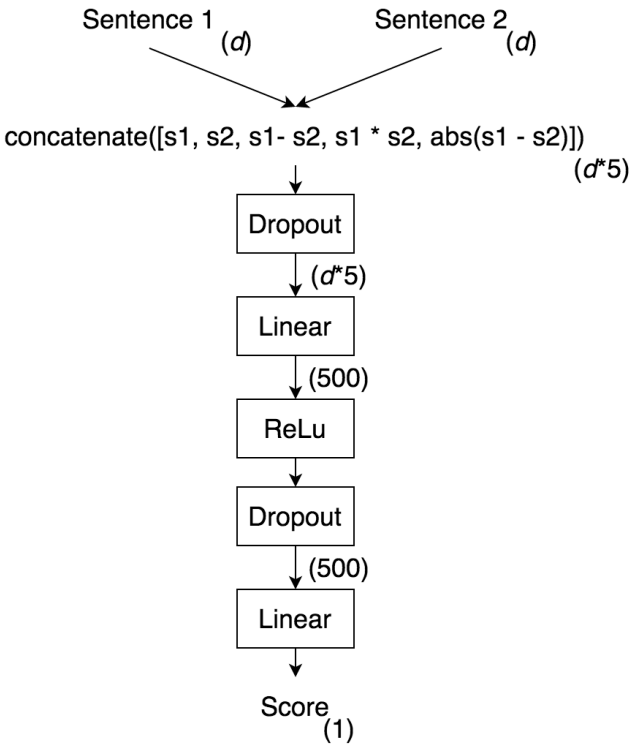

Figure 8.2: P. Xu et al. (2019)'s proposed architecture based on the paper's description and the original code.

the margin. This induces the model to pull apart the scores as far as possible. In this way, the model parameters will be updated only for the samples that are harder (by that margin) to distinguish, otherwise the loss will be zero. A margin of 5.0 was used. P. Xu et al. (2019) showed that their model improves when applied to open-domain tasks, compared to previous methods.

8.3.2 Local coherence for spoken discourse

Patil et al. (2020) proposed to extend P. Xu et al. (2019) by also incorporating audio embeddings. They aimed to use local coherence models to score coherence in spoken discourse rather than written, and such they hypothesized that including speech embeddings (wav2vec) can improve the accuracy of the models. This would be expected as it is known that, for example, prosody plays an important role in discourse structure (Cole, 2015). In this thesis, we will use the Patil et al. (2020) extension of P. Xu et al. (2019) for our experiments. They trained their local coherence model with nine speakers from the IBM Debater data set (Mirkin et al., 2018)[3]. To evaluate the models, they compared the scores the model gives to natural and synthetic speech. The synthetic speech was not obtained by training a TTS system with the IBM Debater speakers but rather they use the Microsoft API to synthesize held-out monologue text from the IBM Debater data set, for a new male and female speaker. To our knowledge, the Microsoft TTS system is not conditioned to context.

To combine the audio embeddings together with the text embeddings into the

[3]They do not make any reference though to the inconsistent recording conditions we found when using this data set for our experiments in Chapter 7.

P. Xu et al. (2019) architecture they use a Bi-linear layer. A Bi-linear layer is similar to a linear layer but it allows the input of two vectors (which do not need to have the same dimension), and aggregates the two vectors into a single output vector. For each sentence the aggregated representation is used instead of the individual embeddings as input to the concatenation step in Figure 8.2, and the same architecture as in P. Xu et al. (2019) is applied.

They compared three experimental conditions: using text embeddings alone, using audio embeddings alone and their approach aggregating both. They saw that when using their models to score held-out natural speech, the models using audio provide wider relative differences between positive and negative scores, the best model being the one that combines both text and audio. They interpreted this as that their discriminative power is higher.

These results were not replicated when testing the models on synthetic speech: for one of the two synthetic speakers only the model combining text and audio was better than the one using audio alone, with a smaller relative difference than for the natural data. For the other speaker, the model using text alone was found best. It is hard to interpret such results as, as mentioned before, the synthetic speech speakers were not used to train the local coherence model.

We took this implementation with the aim of training these models with natural speech and evaluate synthetic data. Our evaluation of synthetic speech though, in contrast, will be aimed at comparing multiple TTS systems. Moreover, we will train a local coherence model and test it on speech from the same speakers.

For our implementation we start from P. Xu et al. (2019)'s original repository[4]. We made sure we were able to replicate the results presented by the authors using the available Wall Street Journal data. Then, we extended it to incorporate the method proposed by Patil et al. (2020) with some adjustments. First of all, for the sake of consistency, and because Glove and wav2vec are not state-of-the-art embeddings anymore, we did a preliminary experiment comparing these embeddings to the ones we had been using through-out this thesis, Deep Spectrum and BERT.

Figure 8.3 shows our implementation. We used one Bi-linear layer instance only. Because this layer is very slow to train and because the dimension of our embeddings is larger than the ones used in the original papers, we first project each into a 512 dimension (using independent linear layers).

To test which set of embeddings is best in this preliminary experiment we used LJ: each chapter was taken as a document, using 35 to train, 5 for validation and 10 for testing. 200 negative and 200 positive samples were extracted per chapter. Sentences were segmented by following the same boundaries used as utterances for the TTS models in previous experiments. We train for 5 epochs and save the model state with the best validation accuracy. For this preliminary experiment only, we present results at the document level, which is the common practice in papers in this field. The models are run to perform inference for each document. Each document is presented in its correct order and then in randomized order, and scores are obtained for each pair of utterances. The average of the scores is calculated and if it is higher for the document in the correct order, then it is considered as a successful, accurate, result. Table 8.1 shows the accuracy of the trained models for the test held-out data. The results show that the text embeddings alone bring better results. When comparing audio embeddings, Deep Spectrum is better than

[4] `https://github.com/BorealisAI/cross_domain_coherence`

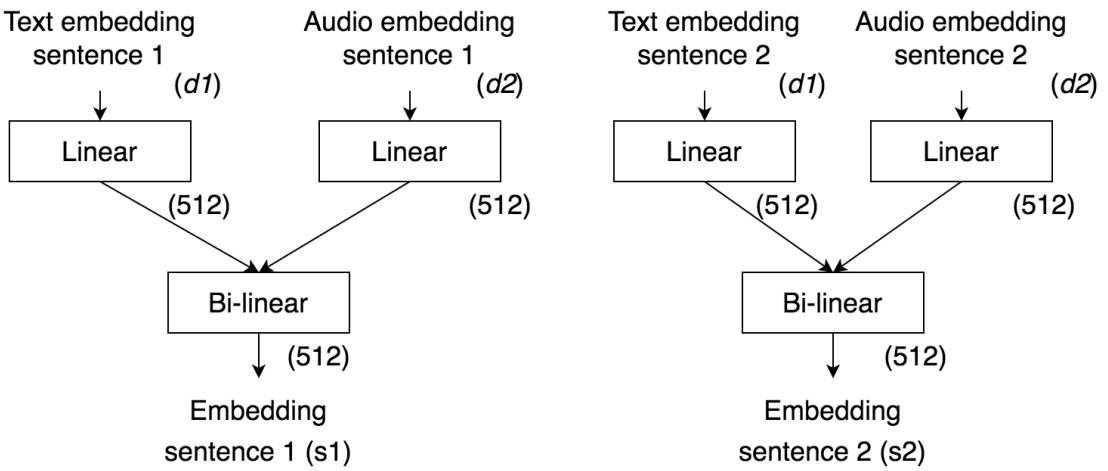

Figure 8.3: Our implementation based on the paper.

wav2vec. Considering these results, we use BERT and Deep Spectrum embeddings for all the next experiments using local coherence models.

Embeddings	**Accuracy**
Glove	1.0
wav2vec	0.52
Glove+wav2vec	0.97
BERT	1.0
DeepSpectrum	0.71
BERT+DeepSpectrum	0.99

Table 8.1: Accuracy over test data for LJ trained coherence models comparing embeddings for text and audio.

8.3.3 Analysing natural speech

We used the local coherence model implementation explained above with our natural speech data to answer **RQ8.3**: *Can we use a local coherence model to* ***analyse natural speech*** *in context and find relevant differences for speaker identity and speaking style?* The models were trained with natural speech and subsequently applied to held-out natural speech to obtain accuracy scores, similarly to the preliminary experiment in the previous section.

We believe that if the test held-out natural speech of a particular speaker obtains high accuracy, it can be understood that sequential pairs of utterances in the natural speech have a stronger relationship, and therefore, are more predictable as a sequence, than randomized pairs. If, in contrast, the accuracy is low, it means that the relationship at the local linguistic context level is hard to distinguish from that of randomized pairs of utterances, implying a weak relationship between sequential utterances in the natural speech.

We do not have a way to confirm that natural speech with higher local coherence scores would correlate with listeners' perception in-context. Ideally, we could have had a "golden speaker", such as a highly proficient orator, for whom we would be

certain that their speaking style has a strong relationship at the local linguistic context level. We could have compared scores for the speakers in this thesis with such speaker.

As an alternative, we propose to compare as many speakers as possible and see if differences exist. Moreover, we already know that LJ was not considered highly natural in-context in the listening test in Section 8.2, and this can be taken as a reference for a speaker for whom we would expect a low accuracy score.

Additionally to **RQ8.3**, we experimented with two variables as this is the first time we trained these models for this purpose: first, considering that we have multiple speakers and data sets, we tested at which "level" the models obtain higher accuracy scores. We tested three levels: speaker-dependent models, data set-dependent models (combining the parallel speakers into one data set, and the podcast speakers into another) and all together. Second, as Patil et al. (2020) did, we trained the models on text-only, audio-only or by aggregating (fusing) the embeddings. Recall that these local coherence models are in no way involved in the TTS training, and that in this subsection, we are training and testing them on natural speech only.

8.3.3.1 Trained systems

We trained the local coherence systems with the same data sets used for TTS. We used the natural speech data of the speakers for which we have trained TTS systems and obtained listening test results in past chapters: LJ, the speakers from the Spotify Podcasts (from Section 7.3, the large versions), and the three parallel speakers from the Parallel Audiobook data set (from Section 7.2). We used the same utterance segmentation we used for TTS in previous chapters, and we vocoded all speech. From the training set we held-out a portion similar in size to the test set for validation (see details on the test sets in the next subsection). We trained every model for 5 epochs, but saving the model state with the best validation accuracy. We extracted utterance-level BERT and Deep Spectrum embeddings for every utterance.

We take as "documents" either chapters for the audiobook data (LJ and the parallel speakers), or episodes for the podcast data. Because our data sets have differing average document length, we sample a different number of triplets per document depending on the data set (extracting as many as possible per speaker). A triplet consists of the current utterance, the correct context and the randomly selected one. Table 8.2 shows the number of training triplets extracted per speaker.

We trained one system per each of the three levels of data levels described above: **speaker**-dependent models, **data set**-dependent where we simply combine the parallel speakers in one group and the podcast speakers in one group (we leave LJ alone) and, a final **all** models that we train with all the data. For every level we also trained a different feature combination: **audio only**, **text only** and combined, which we called **fused** systems. Therefore, we trained 27 systems.

8.3.3.2 Testing and analysis

We tested the systems with the held-out test data for every speaker in turn. Unlike for the preliminary experiment, where we presented accuracy scores averaged at the document level, in this section we present accuracy scores averaged for every triplet. We believe this is more appropriate for us, as we are concerned about the local linguistic context, and not about an entire document. To obtain these scores then,

Data set	**Training triplets**
LJ	15600
ekl	3400
sde	3400
mth	3400
Show 1	16100
Show 2	13000

Table 8.2: Training triplets for local coherence models per speaker.

the tested documents are turned into triplets. If the model gives a higher score to the pair that includes the positive sample than to the negative one, it is considered correct. The test sets match the chapters/episodes used to sample test utterances for the TTS systems: for LJ we use chapters 23 and 24 (∼250 utterances), for the speakers from the Parallel Audiobook dataset chapters 1 and 4 (∼200 utterances), for Show 1, two episodes (∼200 utterances) and for Show 2, one episode (∼250 utterances). We always test each speaker independently.

We present four different tables with the accuracy scores. Table 8.3 reports accuracy scores using audio-only embeddings for all levels, Table 8.4 reports accuracy scores using text-only embeddings for all levels, Table 8.5 using both, combined embeddings, and Table 8.6 reports scores averaging levels and embedding type, in order to facilitate the discussion further below. Accuracy scores that are significantly better than chance (0.5) are signaled with an asterisk symbol, and statistical significance was tested using the Binomial test. Notice that some results are merged: for LJ there is no data set level scores (only one speaker in the data set), and for the text-only condition the scores are the same for all the speakers from the Parallel Audiobook data set.

Recall that these scores correspond to **accuracy** (from 0 to 1) and that both the test and the training data in this section are **natural speech**. A high accuracy score can be interpreted as that the model is able to easily predict if a pair of held-out utterances is an authentic ordered pair or not. In the case of text, this is interpreted as a highly coherent text. When including speech (audio embeddings), we believe this can be interpreted as that there is a strong acoustic predictability between the two utterances.

Next we discuss the results, with respect to **RQ8.3** and to the experimental variables we tested (level and embedding type). We are aware that we cannot make a direct link between these accuracy scores, which are given to natural speech, with listening test results in previous chapters for the synthetic speech from models trained with this data with our method. Nevertheless, we find interesting patterns.

(1) *Are there any important differences in accuracy scores depending on which data level we train?* If we consider the average results per level (Table 8.6, bottom row), we see that there are no major differences between the levels, with the models trained with all the data giving slightly better overall performance. When considered together with the embedding type used, we see that only for the **fused** condition there is a gain from **speaker**-level to the **data set**-level. It is very likely that learning the fusion of the embeddings requires more data for better results, particularly, for the speakers with the smallest data. This can be seen in Table 8.5, where the models

	Level		
Speaker tested	**Speaker**	**Data set**	**All**
LJ	0.50		0.47
ekl	0.53	0.50	0.50
sde	0.51	0.57	0.57
mth	0.57	0.57	0.56
Show 1	0.58*	0.55	0.59*
Show 2	0.49	0.52	0.50

Table 8.3: Audio-only local coherence models accuracy scores.

	Level		
Speaker tested	**Speaker**	**Data set**	**All**
LJ	0.65*		0.65*
ekl	0.74*		0.80*
sde			
mth			
Show 1	0.78*	0.80*	0.77*
Show 2	0.64*	0.66*	0.65*

Table 8.4: Text-only local coherence models accuracy scores

trained at the speaker-level for two of the parallel speakers were not better than chance. Interestingly, "mth" which has the same amount of utterances as "ekl" and "sde" was better than chance. This might point to an interaction between amount of data per speaker and speaker identity (recall Table 8.2).

(2) *Are there any important differences in accuracy scores depending on the type of embeddings?* Table 8.6 clearly shows, when considering the average column, that the **text**-only models are superior, which can be also seen in Table 8.4, where all the models are better than chance for every test performed. This matches our preliminary results in Section 8.3.2. Crucially, the audio-only condition is very hard for the local coherence models, with an average accuracy of only 0.53. We can see in Table 8.3 that most of the models are *not* better than chance, except for Show 1 from the Spotify Podcast data.

(3) *Are there any important differences in accuracy scores depending on the speaker identity?* We can see that the accuracy scores given by the local coherence models reinforce the speaker variability that we have seen in previous chapters. For example, in Table 8.3 when considering audio-only embeddings, only one speaker obtained scores significantly better than chance. We do not believe this is due to the amount of data available, as LJ had a very similar number of triplets in the training data, and even when testing with the **all** system, this difference is replicated. Moreover, the speakers from the Parallel Audiobook data set, for which the text data is exactly the same, showed differences in the fused system, where only for one of them ("mth") scores significantly better than chance were obtained at the **speaker** level. With regard to **RQ8.3** then, we can confirm that these models can be used to find significant differences among speakers when analysing their natural speech.

	Level		
Speaker tested	**Speaker**	**Data set**	**All**
LJ	0.68*		0.66*
ekl	0.61	0.75*	0.75*
sde	0.58	0.75*	0.74*
mth	0.64*	0.68*	0.71*
Show 1	0.76*	0.77*	0.75*
Show 2	0.63*	0.68*	0.61*

Table 8.5: Fused local coherence models accuracy scores

	Level			
Embedding	**Speaker**	**Data set**	**All**	Average
Audio-only	0.53	0.53	0.53	0.53
Text-only	0.71	0.72	0.74	0.72
Fused	0.65	0.71	0.70	0.69
Average	0.63	0.65	0.66	

Table 8.6: Level/Embedding averaging all speakers accuracy scores

(4) *Are there any important differences in accuracy scores depending on speaking style (audiobook vs podcast data)*? We do not see a clear difference in terms of accuracy scores between different styles. However, we know from J. Li and Jurafsky (2016) that domain is an important variable for coherence models. Recall that the base architecture of these local coherence models by P. Xu et al. (2019) is measuring the distance between the representations of the two utterances in the positive and negative pairs such that, for example, if we consider the text-only condition, we can assume that a smaller domain will obtain better scores: a more limited vocabulary would be seen, increasing the similarity between the sentences. We believe that this is why for the **text** only system with **all** data, the speakers from the Parallel Audiobook data set had the best results: the data corresponds to one book only. In contrast, LJ is a collection of chapters from different books of diverse nature. A similar pattern can be found for the speakers of the Spotify Podcasts: Show 1 is about *one* video game only (more limited domain, better scores), while Show 2 is about video games in general. Podcasts also tend to have a structure that repeats for every episode.

Regarding the results and the discussion above, especially the statistical significance of the accuracy scores obtained, we believe that the local coherence models are providing reliable and interesting results for the analysis of natural speech. Therefore, we proceed in the next section to apply these models to synthetic speech.

8.3.4 Evaluating synthetic speech

We evaluate synthetic speech using the *same* local coherence systems trained with vocoded natural speech. Although we know that there is a mismatch between the training and testing data, we believe that the synthetic speech that behaves most

similarly to the natural speech will obtain better scores from these systems. Recall that all natural speech to train the local coherence systems in the previous section was vocoded in order to make audio quality more similar.

Therefore, in this section we seek to answer the last research question of this chapter, **RQ8.4**, where we test if we can use the local coherence models trained with natural speech to evaluate synthetic speech in-context, as a proxy for an in-context listening test. The motivation for this experiment is to see if we can have a quick method to evaluate synthetic speech in-context while the field makes further progress on fine-tuning the in-context listening test design.

This experiment can be categorized as an objective evaluation, as described in Section 2.5. The method we will describe also shares some similarities with the recent efforts on automating the evaluation of synthetic speech (Patton et al., 2016; Lo et al., 2019; Williams et al., 2020; Leng et al., 2021), but these approaches are fundamentally different to what we propose here: the trend there is to learn to predict listening test scores through supervised training of models based on real scores given by listeners. Even if we wanted to try such an approach, we do not have a substantial amount of in-context listening test results.

Through the local coherence models we expect to evaluate a very specific dimension of the synthetic speech, which is if there are predictable sequential acoustic patterns. In this sense, the proposed objective evaluation is more limited than a full listening test (or than trying to predict listening test scores), where people judge multiple aspects of speech. But we believe that this is precisely what we need to test to reach some conclusion regarding whether our proposed method is working or not, and if it is working, that it is actually doing what we designed it for.

Because we do not have in-context listening test scores for all speakers, we cannot test to see if there is a correlation with such scores and the results, that we will shortly present, when applying the local coherence models to evaluate synthetic speech in-context. It is actually possible that they do not correlate, and that increasing the similarity to the ground-truth speech local linguistic context might be considered less natural.

Nevertheless, this objective evaluation might give evidence whether our method to condition a TTS system on context is improving or not in generating a current utterance that depends on the previous one. If that is the case, the synthetic speech generated with our method should obtain a higher score than baseline synthetic speech. In contrast, if a very low score is given to a pair of synthetic utterances, we will interpret that as either there is no acoustic relation between the sequential utterances, such that it cannot be discriminated from random pairs, or that the speech is so constant, that even when randomized, the acoustic pattern is unchanged.

8.3.4.1 Objective evaluation

Because the local coherence models' scores cannot be taken as absolute values, we designed an objective evaluation that outputs an order for the systems evaluated, including the vocoded speech. We believe that the systems with scores closer to those of the vocoded speech are more similar in speech characteristics when taken as pairs of utterances. The synthetic speech of a given TTS system is grouped in pairs of utterances, an then are run through the local coherence model, which outputs a coherence score for every pair. Scores are averaged across each system, and then the systems are ordered by the average score. We will consider the best system as

the one with the most similar average score to the vocoded speech. Because these scores correspond to the ranking function that the local coherence model learns, it is not clear how to test for statistically significant differences, and as such we prefer to focus only on the relative order of the systems with respect to the vocoded speech.

To evaluate the synthetic speech we will use the **fused** local coherence models trained in the previous section. Because our main goal is to test the sequential similarity of the speech, we obviously cannot use the text-only based model. Considering the very low accuracy scores for the local coherence model using audio-only, the fused-based model seems to be the best choice. Regarding which model to use in terms of level, we decided to use the speaker-dependent models. It is not yet clear how the function of the local coherence models trained with multiple speakers combines their characteristics.

Furthermore, we decided to test the in-context objective evaluation for only three of the speakers for whom we had trained local coherence models: LJ, mth and Show 1. We selected these three speakers because for all of them the fused local coherence model at the speaker level obtained accuracy scores significantly better than chance. Although this was also the case for Show 2, we preferred to discard it as it was not clear how the inconsistent recording conditions might affect the objective evaluation. We used the exact same utterances as test samples as the ones used to test the accuracy of the local coherence models for the natural speech. For Show 1 we include below further comparisons regarding data size, by comparing the small, mid and large TTS models trained in Section 7.3.

Table 8.7 lists the systems compared with the objective in-context evaluation. For all speakers we compare the systems in the top half. For LJ only, we also additionally compare the systems using our method to condition a TTS system on delexicalized acoustic context. This is why for the first half of the systems we differentiate them with the "full" method (using DS-utt + BERT-word). Moreover, we also compare the use of ground-truth or synthetic context for the delexicalized variant of our method, including when we manipulated the fundamental frequency for the latter.

We have three hypotheses regarding what we expect to see in the results when evaluating synthetic speech in-context with the local coherence models for all speakers. The first hypothesis is a sanity check that our proposed objective evaluation provides a reasonable way to compare speech: we expect that natural vocoded speech should always obtain the best scores. The second and third hypotheses are concerned with comparing our method to condition TTS on context with a baseline, and to compare the use of ground-truth or synthetic context as acoustic context. We also have a final hypothesis for systems using the variation of our method with delexicalized acoustic context, particular to LJ.

- **H1**: vocoded natural speech, which matches the characteristics of the training data for the local coherence models, should obtain higher scores than any synthetized speech.

- **H2**: synthetic speech generated by our full method to condition TTS systems, when using **ground-truth** acoustic context, should obtain local coherence scores closer to the vocoded speech in comparison to the local coherence scores for the baseline generated speech.

	System	Label
LJ, mth, Show 1	Vocoded natural speech	*voc*
	Baseline	*base*
	Full context conditioning method, using ground-truth acoustic context	*full-gt-ctxt*
	Full context conditioning method, using synthetic acoustic context	*full-synth-ctxt*
LJ only	Delexicalized context conditioning method, using ground-truth acoustic context	*delex-gt-ctxt*
	Delexicalized context conditioning method, using synthetic acoustic context	*delex-synth-ctxt*
	Delexicalized context conditioning method, using synthetic acoustic context and fundamental frequency manipulation	*delex-synth-ctxt+manip*

Table 8.7: Systems tested with the in-context objetive evaluation.

- **H3**: synthetic speech generated by our methods to condition TTS systems, when using **ground-truth** context, should obtain local coherence scores closer to the vocoded speech in comparison to the local coherence scores for synthetic speech generated by our methods using **synthetic** acoustic context.

- **H4**: (for LJ only) considering that the full method leverages more detailed acoustic information during training: the synthetic speech generated by our method using ground-truth delexicalized context should obtain local coherence scores lower than those of the full method using ground-truth context. Conversely, the speech generated by our method using synthetic delexicalized context should obtain local coherence scores higher than those of the full method using synthetic acoustic context.

8.3.4.2 Results

Next we discuss the results with respect to the hypotheses. As stated in the previous section, instead of taking the scores as absolute values, we limit our discussion to interpreting the order in which the systems can be located to with respect to the vocoded speech local coherence scores (see Section 8.3.5 for further discussion). Results are presented in Tables 8.8, 8.9 and 8.10.

For all speakers we find evidence supporting **H1**: the vocoded speech (*voc*) was scored the highest. This is encouraging evidence that our proposed objective evaluation is correctly capturing the in-context behaviour of natural speech. We obtained mixed evidence for **H2**: for LJ, the scores given to the baseline (*base*) speech were the closest to the vocoded speech scores. In contrast, for mth and Show 1 (for all TTS system sizes) our full method using ground-truth acoustic context (*full-gt-ctxt*) was the closest to the vocoded speech scores.

full-synth-ctxt	*delex-synth-ctxt*	*delex-synth-ctxt+manip*	*delex-gt-ctxt*	*full-gt-ctxt*	*base*	*voc*
-3.1	-3.05	-2.96	-2.93	-2.53	-2.42	-2.18

Table 8.8: In-context objective evaluation scores per system for LJ.

full-synth-ctxt	*base*	*full-gt-ctxt*	*voc*
-17.85	-12.99	-12.85	-10.27

Table 8.9: In-context objective evaluation scores per system for mth.

TTS size	*full-synth-ctxt*	*base*	*full-gt-ctxt*	*voc*
Large	-3.44	-2.44	-1.07	-0.71
Mid	-3.30	-2.91	-0.84	
Small	-3.55	-2.50	-1.18	

Table 8.10: In-context objective evaluation scores per system for Show 1.

For all speakers, including the additional systems for LJ and the different sized TTS systems for Show 1, we find evidence to support **H3**: the use of ground-truth acoustic context (*full-gt-ctxt*) always obtains higher scores than the use of synthetic acoustic context (*full-synth-ctxt*) for our proposed method, including *delex-synth-ctxt* < *delex-gt-ctxt*. This confirms that there is relevant information lost during inference as a consequence of exposure bias when using our proposed method to condition on acoustic context.

Finally, we find evidence to support **H4**, regarding the use of delexicalized speech with our method: as this condition does not use text and, furthermore, encodes a filtered amount of acoustic context, it obtains lower scores when using the ground-truth than the full context method (*delex-gt-ctxt* < *full-gt-ctxt*). But, at the same time, we find evidence to support our impressions that this system can bridge the exposure bias gap (at least for LJ), as *full-synth-ctxt* < *delex-synth-ctxt*. Moreover, the fundamental frequency manipulation further lessens the gap as *delex-synth-ctxt* < *delex-synth-ctxt+manip* < *delex-gt-ctxt*.

8.3.5 Discussion

According to the results of our objective evaluation, we conclude that our proposed method to condition a TTS system on context can generate synthetic speech with more similar characteristics to natural speech, when evaluating pairs of utterances. We observe this result for two of the three speakers evaluated, the exception being LJ. It seems that speaker identity is the main cause for such differing results. Training data size (for TTS) does not seem to be related, as we tested the three different sized TTS systems for Show 1, and all of them presented the same pattern (recall that the mid-sized TTS system for Show 1 was trained with relatively the same number of hours of speech as LJ).

This means that for Show 1 and mth our proposed full method, using ground-truth context, is better at reproducing the sequential behaviour in the speech than the baseline. This could be evidence that the speech variation between pairs of utterances is more predictable for these speakers than for LJ. For LJ, in contrast, the baseline has in-context speech characteristics closer to natural speech, however our method does not capture these characteristics so well. Although we would need more information to confirm a pattern, we see a trend with the predictability of pairs of utterances in the natural data. Recall that mth and Show 1 obtained the highest accuracy scores when applying the audio-only local coherence model to natural speech, while LJ had some of the lowest scores (Table 8.3).

The results in this chapter, both from the listening test and the objective evaluation, seem to point to a certain particularity of LJ's speech in context. In the listening test, the natural speech in context was not considered significantly more natural than any of the synthetic speech. With our objective evaluation, our method does not seem to be able to capture sequential speech behaviour as well as the baseline, which does not leverage contextual information. How is it possible then that our method brought significant improvements when evaluating isolated utterances in Chapters 4 and 5? It could be that the method is learning something other than the local context relationships we were expecting to capture for this speaker. The LJ chapters tested were also different, and that might have an effect too. Additionally, other authors have also used LJ to try to improve synthetic speech by conditioning on context, and have also found improvements evaluating isolated utterances (G. Xu et al., 2020; Y. Li et al., 2022), although using text context only. G. Xu et al. (2020) also raised concerns about LJ's style, regarding it as not being the most appropriate for such experiments. The evidence we present here further supports this idea.

Nevertheless, we believe that this does not invalidate our results for LJ in Chapters 4 and 5. Our proposed method is indeed learning something from the local context, but not the sequential behaviour that we were expecting to capture, likely because this behaviour is not easy to predict from LJ's natural speech in context. Moreover, we believe that it is not methodologically appropriate to make connections between the listening tests performed in this thesis for isolated utterances and the objective in-context evaluation just presented. Firstly, as shown by Clark et al. (2019), listening test scores for isolated utterances cannot be used to predict in context ones (although for utterances in the paragraph context). Secondly, because as we said before, our in-context objective evaluation is only limited to score the similarity of pairs of utterances for synthetic speech with respect to the natural speech of the same speaker. Whether this correlates with perceived overall naturalness, either for isolated or in-context utterances, we still do not know.

A clear result is that regardless of whether isolated utterances are evaluated through listening tests, or utterances are evaluated in context through our objective evaluation, speech generated with our proposed method using synthetic acoustic context performs worst. We confirm then that exposure bias is a problem to be accounted for if our method is to be applied using synthetic acoustic context. However, this is something particular to our method, which can be improved but is not our focus. It is more important to have shown evidence that the use of ground-truth context, through our method, can generate synthetic speech that better approximates natural speech in context, as per our objective evaluation, at least for some speakers. Moreover, this shows that our objective in-context evaluation makes predictions that are reasonable, and that this evaluation could be used in the future for quick development of methods to incorporate context into TTS.

It would have been ideal to test Show 1 and mth with the in-context listening test design used in the first half of this chapter (Section 8.2) to see if we could obtain results that matched the scores when applying the local coherence models to synthetic speech. However, because we were not certain whether the design of the in context listening test was reliable, or whether LJ was the problem, we decided to invest our remaining research time we into the objective evaluation.

Nevertheless, by considering the results and this discussion, the local coherence models trained with natural speech, when applied to synthetic speech, can be taken as a proxy for an in-context evaluation of synthetic speech (answering **RQ8.4**). We highlight once more that we believe at the moment the best way of interpreting these scores is by solely comparing the relative order of the systems with respect to the vocoded speech. Recall that we cannot compare absolute scores nor relative differences across speakers, because all scores were obtained with speaker-dependent local coherence models. Moreover, although it is tempting to compare for a single speaker the differences between average scores per system, we do not what constitutes a significant difference. We provide the average scores per system nevertheless for transparency. The evaluation presented here consists only of the first steps towards evaluating a dimension of speech in context that has not been evaluated before through objective means. There is still room for improvement, such as seeing if differences in scores correspond to statistically significant perceived differences, once an in-context evaluation has been standardized.

8.4 Conclusions

This chapter, the last experimental chapter of this thesis, dealt with the problem of in-context evaluation of synthetic speech. In the first set of experiments of this chapter, Section 8.2, we applied the in-context listening test design first proposed by (Clark et al., 2019) and then extended by us (O'Mahony et al., 2021) to evaluate our proposed method to condition TTS systems on context, a baseline and the ground-truth speech for LJ.

Research questions **RQ8.1** and **RQ8.2** were no longer pertinent after we realized that the results contradicted what had been seen by Clark et al. (2019) and O'Mahony et al. (2021), and our own expectations. The vocoded natural speech was not considered significantly more natural than either of the synthetic speech options, and there were no significant differences between synthetic speech generated by our method and the baseline. We thought of three possible reasons that could explain

such results: the LJ natural speech is not considered natural by the listeners in context; we cannot obtain significant improvements through our proposed method when using synthetic acoustic context (in contrast to using ground-truth acoustic context, even when using the delexicalized speech); and, the in-context listening test design requires further development.

To provide evidence to support or reject these possible reasons and to further explain the speaker variability results in Chapter 7, the second set of experiments in this chapter were proposed to use local coherence models. We used them in two consecutive experiments: first, training them with the natural speech of multiple speakers and obtaining accuracy scores for held-out natural speech; and second, applying the same models to score synthetic speech and compare TTS systems with respect to the vocoded speech scores, an evaluation of the synthetic speech as a proxy of an in-context listening test. We found significant differences related to speaker identity (answering **RQ8.3**) using the local coherence models for natural speech and that these same models can be applied as an in-context objective evaluation for synthetic speech (answering **RQ8.4**).

We can make further conclusions beyond those research questions. As a first set of conclusions we link the two halves of this chapter. We want to explicitly support or reject the possible reasons given above to explain the in-context listening test results, considering the results presented at both stages with the local coherence models. We conclude that:

(1) it seems harder to capture LJ speech characteristics in context, which we believe can be linked to the lower predictability of pair of utterances when considering speech alone. Crucially, this could be potentially true for other speakers too; (2) we obtain significant improvements through our method only when conditioning the model at inference time with ground-truth acoustic context, but not when using synthetic acoustic context, and only for some speakers; (3) the in-context listening test design seems to be generally correct, and our unexpected results seem to be related to the use of LJ. However, this design still needs to be further standardized.

In relation to the concerns established in Chapter 7, we see that the local coherence models results allow us to conclude that: (4) there are substantial differences between speakers in terms of how predictable the relationship at the local linguistic context level in the natural speech is especially when considering acoustics only, such that it seems that our method can only bring significant improvements in-context (considering objective evaluation) only for speakers with a stronger acoustic relationship at the local linguistic context level.

Finally, from the discussion we additionally conclude that: (5) the local coherence models applied as an objective evaluation to the synthetic speech can be used as a proxy for in-context evaluation. They can be used for rapid development of further methods; (6) the local coherence models applied to the natural held-out data can be used to find speaker differences. Moreover, it could be possible to apply them in order to select better data sets that will be more useful for research on the use of context to improve TTS; and finally, from the analysis of the ground-truth data, (7) learning acoustic context relationships is harder than learning textual context relationships.

8.5 Chapter contribution

This last experimental chapter belongs to Part III of this thesis. The main goal was to confirm if our method to condition a TTS system on context does indeed improve synthetic speech when evaluated in-context. At the same time we managed, through the use of the local coherence models, to understand better what the necessary conditions are for our method to bring improvements.

With respect to the next steps stated in Section 4.6 after the prototype, we have now provided further and definitive evidence with respect to the effect of "Synthesized acoustic context" and "Data domain and speaker". Finally, we have made a meaningful contribution to the topic of "In-context evaluation", by collaborating with developing further the in-context listening test design (O'Mahony et al., 2021). Moreover, we also proposed, and showed supporting evidence for, the use of local coherence models as a proxy for in-context evaluation. Although Patil et al. (2020) proposed something similar first (on which we base our experiment), we are the first to propose the use of these models to compare several speech systems as an in-context evaluation.

Finally, the results and conclusions obtained in this final chapter, in combination with all the results obtained in the previous chapters, gives us enough evidence to assess if the main goals of this thesis have been achieved or not, which we will do in the next chapter. The output obtained in Part III of this thesis, directed at deepening the understanding of the use of context to improve TTS, has allowed us to identify what our proposed method is really doing, its limitations, and which aspects are most promising for future development. In the following Conclusions chapter we not only recapitulate and discuss further the contributions of this thesis but also suggest some guidelines for future research.

Chapter 9

Conclusions

In this final chapter we start in Section 9.1 by explicitly answering the research questions stated in the Introduction (Section 1.3), summarizing the results and contributions reported in this thesis. Tables 9.1 and 9.2 summarize the experiments, results and contributions of every experimental chapter presented in this thesis, for Parts I and II, and Part III, respectively. Next in Section 9.2, we present an up-to-date review of papers that, similarly to us, research the use of context to improve TTS. This review includes papers found up to August 2022. We finish the chapter with Section 9.3, discussing some final remarks that include limitations, open questions and future work.

9.1 Research questions answered

Beginning this thesis, at the Introduction chapter, we proposed the following research questions to guide the work of this thesis:

- **RQ1**: can we improve Text-to-Speech synthetic speech naturalness by conditioning on local linguistic context?
- **RQ2**: would the amount of any such improvement be greater for some speakers/styles than for others?
- **RQ3**: would we find significantly more improvements when evaluating synthetic speech in-context than for isolated utterances?

We claim that we have provided answers to all these questions. Our most important results for **RQ1** are:

- It is possible to improve TTS systems by conditioning them on local linguistic context representations.
- Improvements depend on the representations and the methods used to condition the TTS system.
- Methods that incorporate context for inference only or to extract features from context to augment the **input** to the TTS system did **not** bring improvements.

Our best proposed method brought improvements to synthetic speech naturalness only when conditioning on ground-truth local linguistic context and when evaluating isolated utterances through listening tests.

	Part I: main method		**Part II: other methods**	
	Chapter 4	**Chapter 5**	**Chapter 6**	
Experiments	We proposed a method to condition TTS on acoustic context, testing a multi-task loss. We compared: Acoustic context encoder + no additional task; Acoustic context encoder + Order task; Acoustic context encoder + Next task.	We compared representations from context to condition TTS: Acoustic vs textual features; extracted with jointly-trained vs pre-trained models; utterance vs word-level representations.	We tried to improve inference with TTS+VAE by measuring distance to contextual representations between the current utterance to the training data samples to drive VAE. We compared textual and acoustic context representations.	We analysed podcast data through Centering Theory-based models. We looked for prosodic correlations with extracted features. We trained TTS systems augmenting input sequences with center and transition features.
Results	Systems conditioned on context were significantly more natural. The use of a multi-task loss can improve naturalness. The best method was no longer preferred when listened to in-context.	The best results were obtained by: using both textual and acoustic representations; both extracted with large pre-trained models; acoustic context at the utterance while textual at the word-level.	None of the proposed methods was evidently better than using the VAE centroid for inference, and all yield extremely inconsistent results.	We found some significant relationships between the extracted and prosodic features, depending on speaker. When augmenting the TTS input with the new features there was no significant improvement.
Contributions	Preliminary results showed that conditioning TTS on context can improve synthetic speech. We were the first to propose the use of acoustic context. Our method has already been used by other authors.	We provided a systematic comparison of different context representations to condition TTS on context with significant improvements. We provided insights into the effect of using acoustic context.	In comparison to the results of previous chapters, we saw that using context representations for inference only or using context to extract features to augment the TTS **input** did not bring improvements, in contrast to **conditioning** TTS systems on context. We showed that working with found dialogue podcast speech can be very difficult. We saw interesting differences in results at the speaker level.	

Table 9.1: Summary of results and contributions this thesis. Parts I and II.

	Part III: analysis and evaluation		
	Chapter 7	**Chapter 8**	
Experiments	We trained the best method from Part I with multiple data sets to find its limitations regarding: speaker variability; data size; recording conditions; and utterance segmentation. We evaluated using synthetic acoustic context, unlike before.	We performed an in-context listening test of the our proposed method using delexicalized context to improve on exposure bias. We compared ground-truth LJ, to baseline and context-conditioned TTS systems listened in isolation and in context.	We proposed to use local coherence models to analyse natural speech. We used the same models to evaluate synthetic speech in-context, comparing: vocoded speech, baseline, and our method (synthesizing with ground-truth and with synthetic context).
Results	Our method was never better than the baseline, but we saw that: listening test results were speaker-dependent; our method trained with larger data sets was better; inconsistent recording conditions are encoded by the acoustic context; utterance segmentation was not relevant.	Our method was not significantly better for isolated utterances. Unexpectedly, there were no significant differences between any of the systems in the in-context condition. The in-context vocoded speech was rated significantly lower than the same utterances in isolation.	Natural data showed weak coherence scores for acoustic features only, and scores were speaker-dependent. Our method obtained better scores for some speakers, only when using ground-truth context. The same speakers had higher coherence scores for acoustic features for natural data.
Contributions	We established the limits of our best method. We confirmed that improvements are speaker-dependent. We compared diverse monologue data sets (audiobooks and podcasts) to train TTS systems conditioned on context.	Contrary to our expectations, listening test results showed that ground-truth vocoded speech might not be more natural in-context than in isolation, at least for LJ. We proposed an objective evaluation as a proxy for in-context listening tests through local coherence models. These same models could be used in future to select suitable training data for TTS systems conditioned on context. We showed that our method increases the local context speech similarity only using ground-truth acoustic context.	

Table 9.2: Summary of results and contributions this thesis. Part III.

As we will see in Section 9.2.1, the last of these results is consistent with the trends we find in the related work performed in parallel to ours, where most methods either condition TTS systems directly on context representations, or use them to predict acoustic embeddings for the current sentence. Moreover, it is also consistent with the major trends in the deep learning field of making use of complex representations rather than augmenting the input with further information, as we discussed in Section 2.6 of the Background.

Regarding **RQ2** and **RQ3**, our most important results are:

- Improvements brought by the use of local linguistic context through our method have limitations and are, most importantly, speaker-dependent.
- Ground-truth speech might not be considered more natural when listened to in-context than isolated utterances.
- Only some speakers show a stronger predictability at the local linguistic context level for ground-truth speech when encoded as acoustic features only.
- When evaluating with the proposed objective in-context evaluation, our method provides speech sequences closer to natural speech for some speakers and only when using ground-truth acoustic context.

This second set of results are very interesting because they are consistent with what we saw in our review of linguistic evidence in the Introduction chapter, Section 1.2: patterns found when investigating the role of context to encode linguistic structure and to disambiguate meaning were different depending on speaking style (spontaneous or not) and speaker identity. Our results point to the relevance of speaker identity, although we did not have enough speakers for style to obtain a clear answer for it, but both Show 1 and mth (spontaneous and audiobook styles, respectively) obtained expected results when applying our objective in-context evaluation.

The fact that the type and/or frequency of local linguistic context dependencies in speech are speaker dependent, means that they can be very sparse for some speakers. This is consistent with what we saw in Section 6.6.3, when looking for correlations between prosody and the local coherence analysis through Centering Theory, and what we saw in Section 8.3, when analyzing natural speech with the local coherence models using audio features only. We hypothesize that if these patterns are sparse, then our method proposed to condition TTS systems on context will struggle to bring improvements to the synthetic speech when evaluated through our objective in-context evaluation.

This was a result especially relevant with respect to LJ: although we found improvements when evaluating isolated sentences and synthesizing with ground-truth acoustic context for LJ, these improvements where not seen when synthesizing with synthetic acoustic context and evaluating in-context using the proposed coherence models. We want to highlight the possibility that, exceptionally, LJ does not have particularly natural in-context speech patterns. This is supported by the results of the in-context listening test for the ground-truth speech and by the scores obtained when applying the coherence models to analyse the natural data. LJ is a widely used data set for TTS, the first option in many cases, as can be observed for several architectures that have been first published, developed and tested using LJ. It was

also the first option for us in Part I of this thesis, and we decided to develop our methods using this data set. However, as we were not content with having results for one speaker only, we tested our methods with additional speakers, which, when applying the objective in-context evaluation, gave multiple comparison points that allowed us to find the limitations of the LJ data set with respect to other speakers. We consider that the results presented in this thesis are substantial evidence of the shortcomings of this data set, and we strongly discourage researchers from further using LJ to develop methods to condition TTS on context.

Lastly, we also saw by applying the local coherence models to the natural speech that it seems easier to find local linguistic context relationships for text, and as we will see in Section 9.2.1, most methods proposed by related work prefer modelling text than acoustic representations of context.

As explained in the Introduction chapter of this thesis, the linguistic context is only one of the dimensions of context that affect speech. Moreover, in this thesis we limited the scope to the local context only. Considering all the results summarized so far, we can conclude that there is a limit to how much we can improve synthetic speech through the use of local linguistic context: because the amount of information that can be explained through the local linguistic context is limited. In this thesis, at least, we have the found the optimal conditions in which conditioning TTS systems on local linguistic context can bring improvements: our method performed best for speakers with high predictability at the local linguistic context level when encoded as acoustic features only (as measured by our proposed local coherence models), with consistent recording conditions, and using ground-truth acoustic context.

Finally, this thesis provided interesting results for the proposed research questions and also made several contributions which are summarized in Tables 9.1 and 9.2. We want to highlight three contributions that we consider the most important:

- We were the first to propose the use of acoustic representations from context to condition TTS, and we thoroughly analysed its use and found its limitations: these results can be used to inform future development of methods to condition TTS on local linguistic context.
- We re-purposed the use of local coherence models to analyse natural speech and to evaluate synthetic speech in-context: these models can now be used to identify which data sets would be best to train a TTS system and condition it on context, and as a proxy for in-context evaluation until there is further consensus on in-context listening test designs.
- We demonstrated that the improvements that our method can bring are speaker-dependent (among other limitations). To our knowledge we are the only ones who systematically tested a method to condition TTS on context for many speakers and two speaking styles. If we found these differences regarding the data, it is worth asking if methods proposed by other authors might show similar data-dependencies.

9.2 Text-to-Speech and context: a review

In this section we make an up to date review of papers where context is used to improve TTS, with papers as new as August 2022. We limit the scope of this review

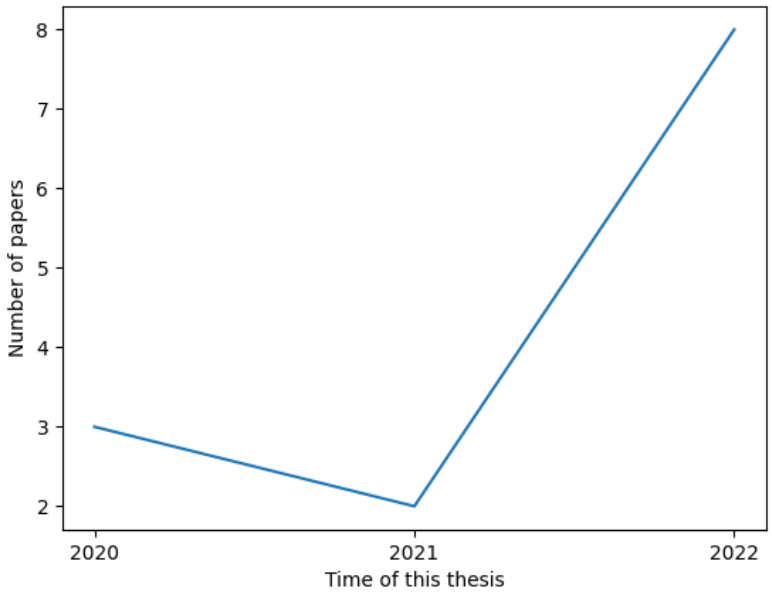

Figure 9.1: Related papers.

to only those papers that propose methods similar to us. Firstly, we noticed that most papers do not define what they mean by context. This can create confusion for the reader. We did our best to determine if the papers were considering context above the utterance or within the utterance, and this review is, of course, focusing on the first group. That means we do not consider within-utterance context methods (Hodari et al., 2021; Karlapati et al., 2021; Morrison et al., 2021; T. Wang, Yi, Fu, Tao, & Wen, 2022) or methods that simply train with inputs that are longer than one utterance, such as chunks or paragraphs (Makarov et al., 2022; Xue, Soong, Zhang, & Xie, 2022). We tried to make an exhaustive search for all the relevant papers for this section, to the best of our knowledge, and considering our research scope. The goal of this section is to paint the big picture of this topic in the TTS field, and see where our work stands among other research, focusing on methods (Section 9.2.1) and then on how these methods have been evaluated (Section 9.2.2).

As we have mentioned before, when we released our first paper (Oplustil-Gallegos & King, 2020) the only paper that had proposed something similar was Tyagi et al. (2020). In parallel to that work, G. Xu et al. (2020) and Guo et al. (2021)[1] published similar papers using only textual representations of context. While these papers were published in 2020, during 2021 only a couple of related papers could be found. In contrast, this year, 2022 we have seen a sharp increase in the interest for these methods, with eight related papers published. Figure 9.1 shows the count of related papers per year. The plot does not include our two papers, only other papers published in those years. As the plot shows, in 2020, while we were developing our method very few people were working on this topic in parallel. We started the prototype of our method in 2020 (Oplustil-Gallegos & King, 2020) and we developed it further in 2021 (Oplustil-Gallegos et al., 2021). During the present year, 2022, we focused on analysis and evaluation, while other people are focusing on developing

[1]Pre-print published in ArXiv by 2020.

further methods. This means that all other work has either been done in **parallel or after our work** about methods (Parts I and II). That is why we are presenting this review in the last chapter of this thesis rather than earlier.

The first thing we notice through this review is that almost every author makes use of sequence-to-sequence architectures as a baseline, the most popular being Tacotron 2 (Tyagi et al., 2020; Guo et al., 2021; P. Xu et al., 2019; Cong et al., 2021) and Fast Speech 2 (Y. Li et al., 2022; J. Li et al., 2022; Nishimura, Saito, Takamichi, Tachibana, & Saruwatari, 2022; N.-Q. Wu, Liu, & Ling, 2022; Lei et al., 2022). We see one paper using VITS (Mitsui et al., 2022) and one with Transformer-based TTS (Y. Wu et al., 2022). Only Yamazaki et al. (2021) uses frame-level DNN systems like the ones described in Section 2.4.1.

We also notice interesting aspects by inspecting the data sets used by these papers, summarized in Table 9.3. Tyagi et al. (2020) is not in this table because their method uses context for inference only: they do not use in-context data for training. Some papers are repeated because they use more than one data set. First, we can see that only 3 languages dominate the research. It would be interesting to expand research to other languages, or see if there language-dependent phenomena related to context. We also notice that 2 papers additional to our work make use of LJ Speech, this is relevant considering the results we saw in Chapter 8 regarding the natural speech characteristics of the LJ speaker when listened to in-context and when analysing it with the local coherence model. The table is organized in two groups: we can see that the first half is using audiobooks and the second half conversational data. As we showed in Chapters 7 and 8, it would be important to explore other types of monologue data than audiobooks, such as podcasts. Interestingly, we notice that many of the authors doing research with conversational data have to record their own data, which is costly (Guo et al., 2021; Cong et al., 2021; Mitsui et al., 2022), but makes TTS training less challenging than when using found conversational data. As we mentioned before, none of the papers reviewed makes a systematic comparison of the results obtained through their methods for different data sets.

9.2.1 Methods and results

In this section we quickly describe the methods proposed by each paper included in this review. Figure 9.2 shows an overview of the papers, grouped according to the type of representations extracted from context. The arrows show which methods have been re-used and by whom in the literature so far. Our papers are in dashed circles. The overview shows interesting trends: the use of only text features is clearly the preferred representation at the moment, followed by the use of text together with acoustic features (use of full mel spectrogram, in contrast to prosodic features, such as f0 countours).

Acoustic only: as mentioned in previous chapters, to our knowledge we were the first to propose such a method in Oplustil-Gallegos and King (2020), which is fully described in Chapter 4. Later, Cong et al. (2021) made used of our proposed method, using the "Next Task" loss. They simplify the regression model for this task to two feed-forward layers only. In their work, they look to model entrainment between speaker for conversational data. In order to achieve that, they extend our method proposed in Oplustil-Gallegos and King (2020) with a Gradient Reversal Layer to remove speaker ID information from the encoding of the previous utterance by the

Paper	**Language**	**Description**	**∼Amount**
Xu et al. (2020) Y. Li et al. (2022)	English	LJ Speech	24 hrs
Xu et al. (2020)	Chinese	Audiobooks	29 hrs
Y. Li et al. (2022)	English	LibriTTS	245 hrs
N. Wu et al. (2022)	Chinese	Audiobooks	200 hrs
Y. Wu et al. (2022)	Chinese	Audiobooks	14 hrs
Lei et al. (2022)	Chinese	Audiobooks	30 hrs
Stephenson et al. (2022)	English	Librivox and Blizzard 2013	66 hrs
Guo et al. (2020)	Chinese	Conversational, tailored	3 hrs
Cong et al. (2021)	Chinese	Conversational, tailored	7 hrs
Yamazaki et al. (2021)	Japanese	Multimodal conversations	56 hrs
J. Li et al. (2022)	English	Conversational, from Youtube	24 hrs
Mitsui et al. (2022)	Japanese	Conversational, free form	18k utts
Nishimura et al. (2022)	Japanese	STUDIES	8 hrs

Table 9.3: Training data sets used by the papers in this review, grouped into audiobooks and dialogue.

GST module. They found improvements with respect to a Tacotron 2 baseline. It is not clear from reading the paper if they were using ground-truth or synthetic acoustic context at synthesis time for the listening tests.

As Figure 9.2 shows, we could not find any other papers proposing to use only acoustic information from the context. Whether this reflects unpublished negative results or the intuition of the researchers, we confirmed in Chapter 8.3 that it is easier to find predictable relationships between sequences of utterances in the text than in the acoustic representations of speech.

Text only: as mentioned above, this corresponds to most of the papers we found. We can further subdivide these methods into three groups: (1) use of context to directly condition training (or the VAE of a TTS models); (2) use of context to learn to predict the output of a reference encoder in a two stage process, where first, the reference encoder learns to extract acoustic embeddings from the current utterance, and then, a predictor is trained to predict those acoustic embeddings through the use of contextual representations; (3) using context to predict different type of labels that are later used as inputs to the TTS model.

Guo et al. (2021) proposes the use of textual context to improve conversational TTS by using a conversational context encoder that processes the chat history up to 10 past sentences. The output of this encoder conditions the TTS system. They find improved naturalness and prosody. P. Xu et al. (2019) investigates the use of textual representations from context for TTS with audiobook data, proposing two different types of encoder to learn the relationship between two past and two future utterances with respect to the current one. They find that their method improves

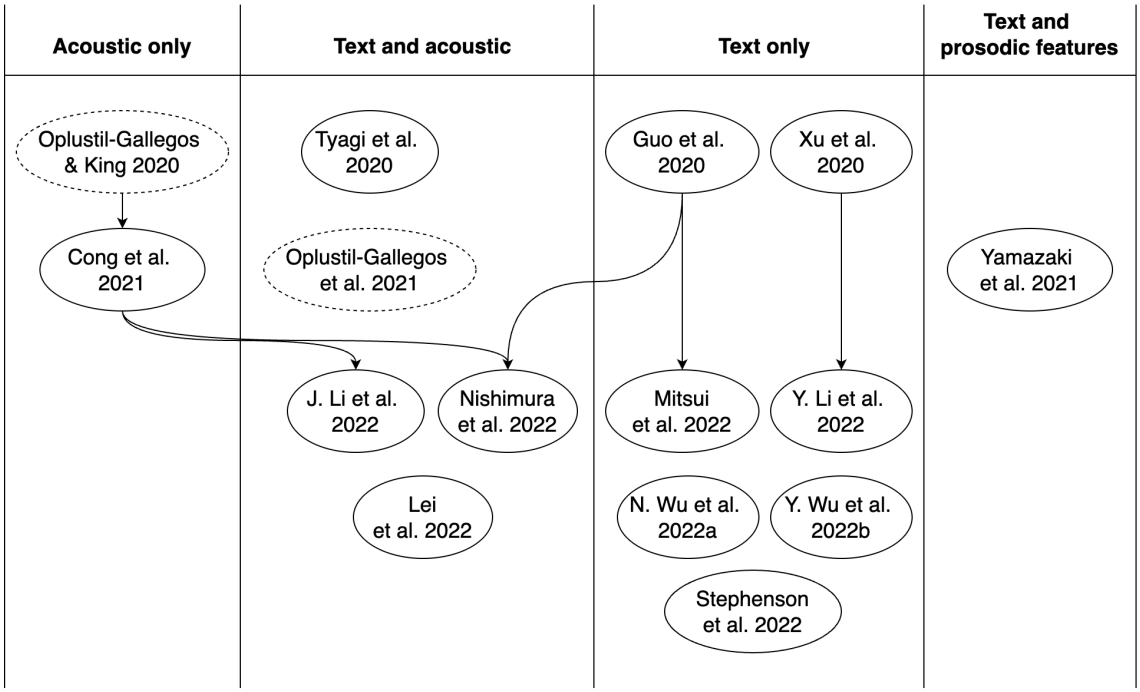

Figure 9.2: Papers by method and re-use of methods from other papers. Our papers are in dashed circles.

duration modelling, naturalness and expressiveness and that they can affect prosody by changing the textual content of the context. The encoder proposed by P. Xu et al. (2019) is then leveraged by Y. Li et al. (2022), to train a conditional VAE together with the TTS architecture. They see that using 5 contextual utterances gives the best results. Their method improves prosody and is preferred in listening tests.

Mitsui et al. (2022) proposes a two stage approach, where first they train a TTS with a VAE to extract acoustic embeddings for the current utterance. Then, textual context representations are extracted following Guo et al. (2021), and are used to train a predictor of the acoustic embeddings. They find that this method improves the naturalness of dialogue speech. N.-Q. Wu et al. (2022) proposes a similar approach. A VAE is trained to encode prosodic features for the current utterance. The output of the VAE is used to train a predictor that takes as input BERT embeddings and phonetic transcriptions of 10 contextual utterances (to the past and to the future). The resulting speech is more expressive than the baseline.

Y. Wu et al. (2022) uses textual representations of one past and future context utterance to train a complex system that predicts emotional labels at the utterance-level to then condition a TTS system. Their proposed model achieves better synthetic speech naturalness, and performs better in a paragraph-level listening test. Stephenson et al. (2022) uses context representations to train a predictor that generates a sequence of prominence labels for the current utterance. They test the use of one or two previous utterances. Unfortunately, they do not find that context improves the accuracy of the predicted prominence sequences.

Text and acoustic: this is the second largest group of papers we found. The first paper to explore the use of both acoustic and textual features from context was Tyagi et al. (2020), which uses distances to the previous utterance to sample from a VAE to drive the TTS during inference. Oplustil-Gallegos et al. (2021) we compare

the use of textual and acoustic features extracted from context to condition TTS training, showing that the combination of both gave the best improvements. These experiments were reported in Chapter 4.

J. Li et al. (2022) proposes a two-stage approach, similar to Mitsui et al. (2022) and N.-Q. Wu et al. (2022), to predict GST embeddings through a graph model with multi-modal contextual features for conversational TTS. Five utterances into the past are encoded through a graph network that takes as input representantions from context that are extracted using the method from Cong et al. (2021) for acoustic features, and BERT for textual representations. The new representations obtained through the graph network are passed through an attention mechanism that finds relevant features to predict the current utterance style embedding.

Nishimura et al. (2022) combined both Cong et al. (2021) and Guo et al. (2021)'s methods for conversational TTS. They propose a cross-modal encoder that aggregates textual and acoustic representations from the context through an attention mechanism, and experiment with four different ways to train it. Their proposed method improves naturalness and speaking-style similarity. It is not clear from reading the paper if they were using ground-truth or synthetic acoustic context when using Cong et al. (2021)'s method.

Lei et al. (2022) proposed a multi-scale style extractor and predictor to condition TTS models on the current sentence, following the two-stage approach. There are three scales: sub-word, sentence and global. Each scale has an encoder for mel spectrograms. The global encoder takes the current sentence and a past and a future one, all together (concatenated). Each encoder then has a style token layer: the outputs of all of them is summed, for a final embedding of the current utterance. To predict that final embedding without the use of mel spectrograms at synthesis time, a hierarchical context encoder that uses BERT embeddings is trained.

Text and prosodic features: Yamazaki et al. (2021) proposes a method to drive their DNN-based (frame-by-frame) system that allows for f0 controllability. They experiment with TTS systems for dialogue agents, also performing language generation. To predict the control signals for the agent turn they propose a prosody generation network that takes as input the word sequence and the fundamental frequency of the user. They saw that their method improved the perceived appropriateness of the synthetic speech in dialogue.

9.2.2 Listening tests

In this section we focus on the listening test designs that were used in the reviewed papers. We briefly included the results in the previous section, as most of these papers report improvements. In this section we are interested to see how many of these papers used in-context evaluation and, if they did, what is reported about the design. We do not consider it a problem if papers did not perform an in-context evaluation, rather, considering our results in Section 8.2 for in-context listening tests, where we highlighted how further work is needed to develop this kind of evaluation, we want to know how much information is provided by researchers regarding the in-context listening test design.

Table 9.4 provides a summary of the type of listening test performed in each paper, the number of listeners and whether they provide in-context listening test results. The papers in the table are organized in two groups, similarly to the previ-

Paper	Type of test	Listeners	In-context?
Tyagi et al. (2020)	MUSHRA	10	Yes
Xu et al. (2020)	MUSHRA Preference test	15	Not clear
Y. Li et al. (2022)	MOS Preference test	23	No
N. Wu et al. (2022)	MOS	13	Not clear
Y. Wu et al. (2022)	MOS CMOS	10	Yes
Lei et al. (2022)	MOS	25	No
Stephenson et al. (2022)	Ranking task	30	No
Guo et al. (2020)	CMOS	20	Yes
Cong et al. (2021)	CMOS	20	Yes
Yamazaki et al. (2021)	CMOS	17	Yes
J. Li et al. (2022)	MOS Preference test	10	Yes
Mitsui et al. (2022)	MOS	30	Yes
Nishimura et al. (2022)	MOS Preference test	25	Not clear

Table 9.4: Summary of evaluations performed in the papers reviewed. For the last column, a "Yes" means that the paper explicitly described at least one in-context listening test (many papers perform more than one test). "Not clear" means that although in the paper they did not explicitly describe the use of in-context samples for the listening test, we found in-context samples in accompanying web pages (for those that provided them). "No" means that, to the best of our understanding, by checking both the paper and samples (if provided), there was no in-context evaluation.

ous table, were the first group performs listening tests for non-conversational data, mostly audiobooks, and the second group for conversational data. A quick inspection shows that papers experimenting with conversational data performed more in-context listening tests.

Now, we focus only on the papers that provided information about in-context listening tests. Particularly, considering our findings on the importance that the instructions have for in-context listening tests (O'Mahony et al., 2021), we review the instructions reported by the papers, and the kind of samples played to participants (if provided). We provide the verbatim instructions given to participants according to the papers.

For papers focused in non-conversational data, Tyagi et al. (2020) asked participants to assess the "suitability of [the] newscaster style" for paragraphs. Y. Wu et al. (2022) asked participants to rate paragraphs, requiring them to "focus on speech style and expressiveness", and to "rate audio considering context which reflects the coherence and appropriateness of the audio's expressiveness". For papers focused on conversational TTS, Guo et al. (2021) played conversation turns one by one for listeners to rate individually first, then played the complete dialogue and asked listeners to rate it again for the overall performance. They asked the question: "is prosodic expression more suitable and more natural with the current context?".

Cong et al. (2021) made participants rate paired utterances, and they report that participants "were asked to compare the naturalness and especially the entrainment of each paired utterance", although it seems unlikely that participants would know the meaning of "entrainment". Yamazaki et al. (2021) also asked participants to evaluate pairs of utterances, asking them to choose "which one was most appropriate in the dialog context". J. Li et al. (2022) evaluated "conversation chunks", but they do not report which instructions were given to participants. Mitsui et al. (2022) evaluated short dialogues of six utterances, and asked participants to evaluate the naturalness of the dialogue, "whether natural entrainment occurred, whether the speaking style was suitable for the context, etc.".

The purpose of this brief review was to show that there is no consensus on how to perform in-context listening tests: every paper instructs participants to do different things, and because the focus of the papers is not on evaluation but rather on developing methods, there is no critical analysis of the way in which in-context evaluation has been performed. We believe that there is need for research into an agreed framework to perform in-context evaluation, or at the very least consensus on which aspects of the listening test design should be reported (something like Wester et al. (2015), but for in-context listening tests). Otherwise, the multiplicity of in-context listening tests designs performed makes it hard to know which of these methods is actually better.

Finally, none of these papers propose objective evaluations or proxies for in-context listening test evaluations. Many papers use traditional objective evaluations of speech quality (such as Mel Spectral Distortion, or Fundamental Frequency Error metrics), but none proposes a method to evaluate objectively the predictability of sequences of synthetic utterances, as we did in Section 8.3 with local coherence models.

9.3 Final remarks

We identify two main limitations that restricted the extent of what we could attempt experimentally in this thesis: first the data, and second, computational resources. At the outset we wanted to experiment with dialogue data rather than with monologues, which was the type of data eventually mostly used in this thesis. This was because we considered that many of the linguistic phenomena where the interaction between context and speech is more explicit, as reported in Section 1.2, would more likely be found in dialogue. However, when looking for suitable conversational data sets for experimental work, we could not find any. Notice how, for the research reviewed in Section 9.2, most conversational data sets were specifically recorded or tailored for their research. We did not have the resources to do that. During my internship at Amazon, where I had the chance to work with conversational podcast data, the duration of the internship was not long enough to fully tackle all the challenges that we reported when working with such data in Chapter 6.

When considering the significant effect that sparsity in the acoustic relationships between pairs of utterances could have for our proposed method, we thought of the possibility of a self-supervised pre-training of our acoustic context encoders. Pre-training is a very popular approach these days that frequently delivers significant improvements. Although we had access to massive data sets like the Spotify podcasts to test such approach, we did not have the computational resources to perform an

experiment of this nature within a reasonable time frame. As a reference point, the authors of HuBERT (Hsu et al., 2021), a self-supervised model to extract speech representations, used 128 GPUs to train the model with 60k hours of speech.

Although not a practical limitation per se, we observe that when we started this thesis the topic of improving TTS through context was at a very early stage (as shown retrospectively in Section 9.2). There was no dominant method to condition TTS systems on context, there was no standard in-context evaluation design and there was no agreement on which type of data to use. This is why we took the path we did, by first proposing a method without having a clear in-context evaluation design or insights into the influence that the nature of the data could have. We then investigated the latter two. We have contributed to each of these fundamental points, as can be seen in our three primary research questions, re-stated at the beginning of this chapter, with a better understanding of each of them, as recapitulated in Section 9.1.

There are always open questions which, because of time limitations, we cannot address further, but that we believe are the logical next steps. We highlight the most important three that refer, once again, to the pillars of this thesis: data, evaluation and methods. First, in order to develop in-context evaluation and new methods to condition TTS systems on context even further, we need to find better data. A large speaker search could be made through the use of the local coherence models, as proposed in Section 8.2. For example, a massive data set like the Spotify podcasts can be used to test the use of the local coherence models for data selection. We predict that in a subjective evaluation, speakers with a high accuracy for automatic coherence scores will also be considered more natural when listened to in-context. Second, it is very important to confirm that the results obtained through the local coherence models to objectively evaluate synthetic speech in-context (Section 8.3) correlate with subjective evaluation in-context. This would require further work confirming the best possible design for in-context evaluation, then looking for relationships to the objective evaluation proposed in our work. Third, having found better data and confirming in-context subjective and objective evaluation, further methods can be developed, or existing methods can be compared side-by-side to achieve greater understanding of which bring the most improvement to synthetic speech. Fourth, we do not consider it a disadvantage *per se* that our method only brought improvements when using ground-truth acoustic speech. There might be an application where the ground-truth context is present and can therefore be used such as a dialogue system, where the voice of the user can be taken as the context. We believe there are many possible solutions to the exposure bias: one option could be to exploit only the textual context which is not affected by the exposure bias, which, as we noticed in the field review, is the main trend at the moment. Recall though that in this thesis we explicitly wanted to focus on the use of acoustic context. However, if we still wish to keep the acoustic context we could improve on the exposure bias by using an acoustic representation that does not encode as much detail from it, but rather only what we want to capture (as we did by using delexicalized context, in Section 8.2.2). Developing such a representation will require further research. Finally, exposure bias is a general machine learning problem for which general solutions had been proposed, such as schedule sampling (Bengio et al., 2015). We did not have time to explore these solutions in this thesis as they require non-trivial hyperaparameter searches, but we believe these could potentially

improve the exposure bias issues.

We see a few interesting avenues for future work. In several chapters of this thesis, we made use of models or methods that aggregate textual and acoustic representations, which is sometimes denoted "multi-modal". For example, in Section 6.4 we made use of the method proposed by Rahman et al. (2020) to acoustically shift BERT representations, or in Section 8.3.2, we made use of the method proposed by Patil et al. (2020) to aggregate audio and text for the local coherence models. We see this as a trend in the TTS field, and further research to integrate representations from text and speech would greatly benefit the representation of context. Second, in most of our work in this thesis, and in all of the papers reviewed in Section 9.2, BERT is used as the default text encoder. This is a logical consequence of the incredible results that BERT has brought across NLP and speech. Moreover, the NSP task should be capturing some level of local context. Nevertheless, it would be interesting to closely analyse how much contextual relationship between pairs of sentences BERT can actually encode, and if these are truly encoding semantic knowledge. Even further, it would be good to consider other ways of encoding text that might be more tailored to the specific phenomena seen in the interactions between context and speech, reported in Section 1.2. It is also important to make a larger comparison of which are the best representations of acoustic context. In this thesis we made use of Deep Spectrum because of previous positive results (Oplustil-Gallegos et al., 2020). Unsupervised speech representations is a growing topic of interest in the community. It would be interesting to compare these, especially now that we have proposed our objective evaluation, as it is possible to perform a large scale comparison without the need of costly listening tests.

Although out of the scope we defined for this thesis, there are other research paths that are important to consider in future work. We want to highlight two topics that we believe will be very important: systematically identify the optimal span of context, and decide whether systems should **always** be conditioned on context. In this thesis we limited the scope to the local context, i.e. one neighbouring utterance only, which, by considering the use of acoustic representations, was further limited to one utterance in the past only. While we still believe that this was a necessary limit on the scope of our work, most very recent papers reviewed in Section 9.2 make use of longer spans of context. But more importantly, no work has systematically tested how the length of that span might increase or decrease the improvements obtained by conditioning TTS systems on context. Lastly, considering the linguistic evidence reported in Section 1.2 and the results of this thesis, we believe that, although there is an overall structural redundancy that is encoded in the context, not all utterances will necessary benefit from the information encoded in the context, as this will depend on their semantic and pragmatic meanings and role in the discourse. We believe it would be very important to consider how to **reset** the context considering these linguistic variables, and how that might improve synthetic speech even further.

We claim a valuable contribution to the topic we proposed in this thesis, by investigating methods, the effect of data and in-context evaluation for the use of local linguistic context to improve synthetic speech. We believe that our work can be leveraged by other researchers and that there is plenty to be done to fully explore this topic, with exciting avenues ahead that can further improve the quality of synthetic speech generated by Text-to-Speech systems.

References

Amiriparian, S., Gerczuk, M., Ottl, S., Cummins, N., Freitag, M., Pugachevskiy, S., ... Schuller, B. (2017). Snore sound classification using image-based deep spectrum features. In *Proc. Interspeech.*

Aubin, A., Cervone, A., Watts, O., & King, S. (2019). Improving speech synthesis with discourse relations. In *Proc. Interspeech.*

Baby, A., Vinnaitherthan, S., Adiga, N., Jawale, P., Badam, S., Adavanne, S., & Konjeti, S. (2020). An ASR guided speech intelligibility measure for TTS model selection. In *Proc. Interspeech.*

Baevski, A., Schneider, S., & Auli, M. (2019). VQ-wav2vec: self-supervised learning of discrete speech representations. In *Proc. ICLR.*

Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Proc. NIPS.*

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

Baljekar, P., & Black, A. W. (2016). Utterance selection techniques for TTS systems using found speech. In *Proc. SSW.*

Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. (2015). Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, *28*.

Black, A., Taylor, P., Caley, R., & Clark, R. (1998). *The festival speech synthesis system.*

Boersma, P. (2001). Praat, a system for doing phonetics by computer. *Glot. Int.*, *5*(9).

Boersma, P., et al. (1993). Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound. In *Proc. IFA.*

Bögels, S., & Torreira, F. (2021). Turn-end estimation in conversational turn-taking: The roles of context and prosody. *Discourse Processes*, *58*(10).

Bredin, H., Yin, R., Coria, J. M., Gelly, G., Korshunov, P., Lavechin, M., ... Gill, M.-P. (2020). pyannote.audio: neural building blocks for speaker diarization. In *Proc. ICASSP.*

Bryant, G. A. (2010). Prosodic contrasts in ironic speech. *Discourse Processes*, *47*(7).

Bunt, H. (1994). Context and dialogue control. *Think Quarterly*, *3*(1).

Bunt, H. (2000). Dialogue pragmatics and context specification. *Abduction, Belief and Context in Dialogue. Studies in Computational Pragmatics*.

Chung, Y.-A., Wang, Y., Hsu, W.-N., Zhang, Y., & Skerry-Ryan, R. (2019). Semi-supervised training for improving data efficiency in end-to-end speech synthesis. In *ICASSP.*

Clark, R., Richmond, K., & King, S. (2007). Multisyn: Open-domain unit selection for the festival speech synthesis system. *Speech Communication*, *49*(4).

Clark, R., Silen, H., Kenter, T., & Leith, R. (2019). Evaluating long-form Text-to-Speech: Comparing the ratings of sentences and paragraphs. In *Proc. SSW.*

Clifton, A., Pappu, A., Reddy, S., Yu, Y., Karlgren, J., Carterette, B., & Jones, R. (2020). The Spotify Podcast Dataset. *ArXiv preprint.*

Cole, J. (2015). Prosody in context: a review. *Language, Cognition and Neuroscience*, *vol. 30*(1-2).

Cong, J., Yang, S., Hu, N., Li, G., Xie, L., & Su, D. (2021). Controllable context-aware conversational speech synthesis. In *Proc. Interspeech.*

Cosi, P. (1993). SLAM: segmentation and labelling automatic module. In *Proc. Eurospeech.*

Crystal, D. (2011). *A dictionary of linguistics and phonetics.* John Wiley & Sons.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT.*

Dubiel, M., Halvey, M., Oplustil-Gallegos, P., & King, S. (2020). Persuasive synthetic speech: Voice perception and user behaviour. In *Proc. cui.*

Eyben, F., Buchholz, S., Braunschweiler, N., Latorre, J., Wan, V., Gales, M. J., & Knill, K. (2012). Unsupervised clustering of emotion and voice styles for expressive TTS. In *Proc. ICASSP.*

Farrús, M., Lai, C., & Moore, J. D. (2016). Paragraph-based prosodic cues for speech synthesis applications. In *Proc. Speech Prosody.*

Finkbeiner, R., Meibauer, J., & Schumacher, P. B. (2012). *What is a context?: Linguistic approaches and challenges.* John Benjamins.

Fong, J., Gallegos, P. O.-G., Hodari, Z., & King, S. (2019). Investigating the robustness of sequence-to-sequence Text-to-Speech models to imperfectly-transcribed training data. In *Proc. Interspeech.*

Fox Tree, J. E., & Meijer, P. J. (2000). Untrained speakers' use of prosody in syntactic disambiguation and listeners' interpretations. *Psychological Research*, *63*(1).

Govalkar, P., Fischer, J., Zalkow, F., & Dittmar, C. (2019). A comparison of recent neural vocoders for speech signal reconstruction. In *Proc. SSW.*

Gravano, A., Beňuš, Š., Chávez, H., Hirschberg, J., & Wilcox, L. (2007). On the role of context and prosody in the interpretation of 'okay'. In *Proc. ACL.*

Griffin, D., & Lim, J. (1984). Signal estimation from modified short-time fourier transform. *IEEE Transactions on acoustics, speech, and signal processing*, *32*(2).

Grosz, B., & Hirschberg, J. (1992). Some intonational characteristics of discourse structure. In *Proc. ICSLP.*

Grosz, B. J., Joshi, A. K., & Weinstein, S. (1995). Centering: A framework for modelling the local coherence of discourse. *Computational Linguistics*, *vol. 21*(2).

Guo, H., Zhang, S., Soong, F. K., He, L., & Xie, L. (2021). Conversational end-to-end tts for voice agents. In *Proc. SLT.*

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

Hirschberg, J. (1990). Accent and discourse context: Assigning pitch accent in synthetic speech. In *Proc. AAAI.*

Hodari, Z., Moinet, A., Karlapati, S., Lorenzo-Trueba, J., Merritt, T., Joly, A., ... Drugman, T. (2021). CAMP: A two-stage approach to modelling prosody in context. In *Proc. ICASSP.*

Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., & Mohamed, A. (2021). HuBERT: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *29*.

Hu, Q., Marchi, E., Winarsky, D., Stylianou, Y., Naik, D., & Kajarekar, S. (2019). Neural Text-to-Speech adaptation from low quality public recordings. In *Proc. SSW.*

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3).

Huybrechts, G., Merritt, T., Comini, G., Perz, B., Shah, R., & Lorenzo-Trueba, J. (2021). Low-resource expressive Text-to-Speech using data augmentation. In *Proc. ICASSP.*

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

ITU. (2006). P. 800.1, Mean opinion score (MOS) terminology. *International Telecommunication Union.*

ITU. (2014). Method for the subjective assessment of intermediate quality level of audio systems. *International Telecommunication Union.*

Jadoul, Y., Thompson, B., & de Boer, B. (2018). Introducing Parselmouth: A Python interface to Praat. *Journal of Phonetics*, *71*.

Jain, S., & Wallace, B. C. (2019). Attention is not explanation. In *Proc. ACL.*

Jeon, S., & Strube, M. (2020). Centering-based neural coherence modeling with hierarchical discourse segments. In *Proc. EMNLP.*

Jernite, Y., Bowman, S. R., & Sontag, D. (2017). Discourse-based objectives for fast unsupervised sentence representation learning. *ArXiv preprint*.

Jia, Y., Zen, H., Shen, J., Zhang, Y., & Wu, Y. (2021). PnG BERT: Augmented BERT on phonemes and graphemes for neural TTS. In *Proc. Interspeech.*

Jurafsky, D., & Martin, J. H. (2021). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.*

Karlapati, S., Abbas, A., Hodari, Z., Moinet, A., Joly, A., Karanasou, P., & Drugman, T. (2021). Prosodic representation learning and contextual sampling for neural text-to-speech. In *Proc. ICASSP.*

Karlapati, S., Moinet, A., Joly, A., Klimkov, V., Sáez-Trigueros, D., & Drugman, T. (2020). Copycat: Many-to-many fine-grained prosody transfer for neural text-to-speech. In *Proc. Interspeech.*

Kato, S., Yasuda, Y., Wang, X., Cooper, E., Takaki, S., & Yamagishi, J. (2019). Rakugo speech synthesis using segment-to-segment neural transduction and style tokens—toward speech synthesis for entertaining audiences. In *Proc. Speech Synthesis.*

Kenter, T., Sharma, M. K., & Clark, R. (2020). Improving prosody of RNN-based English Text-to-Speech synthesis by incorporating a BERT model. In *Proc. Interspeech.*

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. In *Proc. ICLR .*

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes.

Kishore, S., & Black, A. (2003). Unit size in unit selection speech synthesis. In *Proc. eurospeech.*

Klimkov, V., Nadolski, A., Moinet, A., Putrycz, B., Barra-Chicote, R., Merritt, T., & Drugman, T. (2017). Phrase Break Prediction for Long-Form Reading TTS: Exploiting Text Structure Information. In *Proc. Interspeech.*

Kominek, J., & Black, A. W. (2004). The CMU Arctic speech databases. In *Proc. SSW.*

Kuligowska, K., Kisielewicz, P., & Włodarz, A. (2018). Speech synthesis systems: disadvantages and limitations. *International Journal of Engineering & Technology*, *7*.

Łańcucki, A. (2021). FastPitch: parallel Text-to-Speech with pitch prediction. In *Proc. ICASSP.*

Lapata, M., Barzilay, R., et al. (2005). Automatic evaluation of text coherence: Models and representations. In *Proc. IJCAI.*

Latorre, J., Yanagisawa, K., Wan, V., Kolluru, B., & Gales, M. J. (2014). Speech intonation for TTS: Study on evaluation methodology. In *Proc. Interspeech.*

Lee, C.-C., Black, M., Katsamanis, A., Lammert, A. C., Baucom, B. R., Christensen, A., ... Narayanan, S. S. (2010). Quantification of prosodic entrainment in affective spontaneous spoken interactions of married couples. In *Proc. Interspeech.*

Lei, S., Zhou, Y., Chen, L., Hu, J., Wu, Z., Kang, S., & Meng, H. (2022). Towards multi-scale speaking style modelling with hierarchical context information for mandarin speech synthesis. In *Proc. Interspeech.*

Leng, Y., Tan, X., Zhao, S., Soong, F., Li, X.-Y., & Qin, T. (2021). MBNET: MOS prediction for synthesized speech with Mean-Bias Network. In *Proc. ICASSP.*

Lenglet, M., Perrotin, O., & Bailly, G. (2021). Impact of segmentation and annotation in french end-to-end synthesis. In *Proc. SSW.*

Levitan, R., & Hirschberg, J. B. (2011). Measuring acoustic-prosodic entrainment with respect to multiple levels and dimensions. In *Proc. Interspeech.*

Li, B., & Zen, H. (2016). Multi-language multi-speaker acoustic modeling for LSTM-RNN based statistical parametric speech synthesis. In *Proc. Interspeech.*

Li, J., & Jurafsky, D. (2016). Neural net models for open-domain discourse coherence. In *Proc. EMNLP.*

Li, J., Meng, Y., Li, C., Wu, Z., Meng, H., Weng, C., & Su, D. (2022). Enhancing speaking styles in conversational text-to-speech synthesis with graph-based multi-modal context modeling. In *Proc. ICASSP.*

Li, Y., Yu, C., Sun, G., Jiang, H., Sun, F., Zu, W., ... Wang, J. (2022). Cross-utterance conditioned VAE for non-autoregressive text-to-speech. In *Proc. ACL.*

Liu, A. T., Yang, S.-w., Chi, P.-H., Hsu, P.-c., & Lee, H.-y. (2020). Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders. In *Proc. ICASSP.*

Lo, C.-C., Fu, S.-W., Huang, W.-C., Wang, X., Yamagishi, J., Tsao, Y., & Wang, H.-M. (2019). MOSnet: Deep learning based objective assessment for voice conversion. In *Proc. Interspeech.*

Lőrincz, B., Stan, A., & Giurgiu, M. (2021). An objective evaluation of the effects of recording conditions and speaker characteristics in multi-speaker deep neural speech synthesis. *Procedia Computer Science*, *vol. 192*.

Makarov, P., Abbas, A., Łajszczak, M., Joly, A., Karlapati, S., Moinet, A., ... Karanasou, P. (2022). Simple and effective multi-sentence tts with expressive and coherent prosody. In *Proc. Interspeech.*

Malisz, Z., Berthelsen, H., Beskow, J., & Gustafson, J. (2017). Controlling prominence realisation in parametric DNN-based speech synthesis. In *Proc. Interspeech.*

Martinez-Lucas, L., Abdelwahab, M., & Busso, C. (2020). The MSP-conversation corpus. In *Proc. Interspeech.*

McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., & Sonderegger, M. (2017). Montreal Forced Aligner: trainable Text-Speech alignment Using Kaldi. In *Proc. Interspeech.*

McCowan, I., Carletta, J., Kraaij, W., Ashby, S., Bourban, S., Flynn, M., ... others (2005). The AMI meeting corpus. In *Proc. international conference on methods and techniques in behavioral research.*

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in Python. In *Proc. SCIPY.*

Mendelson, J., & Aylett, M. P. (2017). Beyond the listening test: An interactive approach to tts evaluation. In *Proc. Interspeech.*

Mirkin, S., Jacovi, M., Lavee, T., Kuo, H.-K., Thomas, S., Sager, L., ... Slonim, N. (2018). A recorded debating dataset. In *Proc. LREC.*

Mitsui, K., Zhao, T., Sawada, K., Hono, Y., Nankaku, Y., & Tokuda, K. (2022). End-to-end Text-to-Speech based on latent representation of speaking styles using spontaneous dialogue. In *Proc. Interspeech.*

Moore, R. K., & Skidmore, L. (2019). On the Use/Misuse of the Term 'Phoneme'. In *Proc. Interspeech.*

Morrison, M., Rencker, L., Jin, Z., Bryan, N. J., Caceres, J.-P., & Pardo, B. (2021). Context-aware prosody correction for text-based speech editing. In *Proc. ICASSP.*

Nishimura, Y., Saito, Y., Takamichi, S., Tachibana, K., & Saruwatari, H. (2022). Acoustic modeling for end-to-end empathetic dialogue speech synthesis using linguistic and prosodic contexts of dialogue history. In *Proc. Interspeech.*

O'Mahony, J., Oplustil-Gallegos, P., Lai, C., & King, S. (2021). Factors affecting the evaluation of synthetic speech in context. In *Proc. SSW.*

Oplustil-Gallegos, P., & King, S. (2020). Using previous acoustic context to improve text-to-speech synthesis. *ArXiv preprint*.

Oplustil-Gallegos, P., O'Mahony, J., & King, S. (2021). Comparing acoustic and textual representations of previous linguistic context for improving Text-to-Speech. In *Proc. SSW.*

Oplustil-Gallegos, P., Williams, J., Rownicka, J., & King, S. (2020). An unsupervised method to select a speaker subset from large multi-speaker speech synthesis datasets. In *Proc. Interspeech.*

Oxford english dictionary. (2000). Oxford University Press.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning

library. In *Advances in neural information processing systems.*

Patil, R., Singla, Y. K., Shah, R. R., Hama, M., & Zimmermann, R. (2020). Towards modelling coherence in spoken discourse. *ArXiv preprint*.

Patton, B., Agiomyrgiannakis, Y., Terry, M., Wilson, K., Saurous, R. A., & Sculley, D. (2016). AutoMOS: learning a non-intrusive assessor of naturalness-of-speech. In *Proc. NIPS.*

Peiró Lilja, À., & Farrús, M. (2018). Paragraph prosodic patterns to enhance Text-to-Speech naturalness. In *Proc. Speech Prosody.*

Pierrehumbert, J., & Hirschberg, J. (1990). The meaning of intonational contours in the interpretation of discourse. *Intentions in communication*, *271*.

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

Prenger, R., Valle, R., & Catanzaro, B. (2019). Waveglow: A flow-based generative network for speech synthesis. In *Proc. ICASSP.*

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., & Manning, C. D. (2020). Stanza: A python natural language processing toolkit for many human languages. In *Proc. ACL.*

Rahman, W., Hasan, M. K., Lee, S., Zadeh, A., Mao, C., Morency, L.-P., & Hoque, E. (2020). Integrating multimodal information in large pretrained transformers. In *Proc. ACL.*

Rendel, A., Fernandez, R., Kons, Z., Rosenberg, A., Hoory, R., & Ramabhadran, B. (2017). Weakly-supervised phrase assignment from text in a speech-synthesis system using noisy labels. In *Proc. Interspeech.*

Ribeiro, M. S. (2018). *Parallel audiobook corpus.* `https://datashare.ed.ac.uk/handle/10283/3217`.

Roberts, C. (2006). Context in dynamic interpretation. In *The Handbook of Pragmatics* (p. 197-220).

Schneider, S., Baevski, A., Collobert, R., & Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. In *Proc. Interspeech.*

Serrano, S., & Smith, N. A. (2019). Is attention interpretable? In *Proc. ACL.*

Shechtman, S., Fernandez, R., & Haws, D. (2021). Supervised and unsupervised approaches for controlling narrow lexical focus in sequence-to-sequence speech synthesis. In *Proc. SLT.*

Shechtman, S., Haws, D., & Fernandez, R. (2021). Stable checkpoint selection and evaluation in sequence to sequence speech synthesis. In *Proc. ICASSP.*

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., ... others (2018). Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions. In *Proc. ICASSP.*

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR.*

Skerry-Ryan, R., Battenberg, E., Xiao, Y., Wang, Y., Stanton, D., Shor, J., ... Saurous, R. A. (2018). Towards end-to-end prosody transfer for expressive speech synthesis with Tacotron. In *Proc. ICML.*

Song, C., Li, J., Zhou, Y., Wu, Z., & Meng, H. (2021). Syntactic representation learning for neural network based TTS with syntactic parse tree traversal. In *Proc. ICASSP.*

Sridhar, V. K. R., Syrdal, A., Conkie, A. D., & Bangalore, S. (2011). Enriching text-to-speech synthesis using automatic dialog act tags. In *Proc. Interspeech.*

Stahlberg, F. (2020). Neural machine translation: A review. *Journal of Artificial Intelligence Research*, *69*.

Stavropoulou, P., & Baltazani, M. (2021). The prosody of correction and contrast. *Journal of Pragmatics*, *171*.

Stephenson, B., Besacier, L., Girin, L., & Hueber, T. (2022). BERT, can HE predict contrastive focus? Predicting and controlling prominence in neural TTS using a language model. *ArXiv preprint*.

Strelec, M., Rohnke, J., Bonafonte, A., Łajszczak, M., & Wood, T. (2021). Discrete acoustic space for an efficient sampling in neural Text-to-Speech. In *Proc. ICASSP*.

Strom, V., Clark, R., & King, S. (2006). Expressive prosody for unit-selection speech synthesis. In *Proc. Interspeech.*

Suni, A., Kakouros, S., Vainio, M., & Šimko, J. (2020). Prosodic prominence and boundaries in sequence-to-sequence speech synthesis. In *Proc. Speech Prosody.*

Székely, É., Henter, G. E., Beskow, J., & Gustafson, J. (2019). Spontaneous conversational speech synthesis from found data. In *Proc. Interspeech.*

Székely, É., Henter, G. E., & Gustafson, J. (2019). Casting to corpus: Segmenting and selecting spontaneous dialogue for TTS with a CNN-LSTM speaker-dependent breath detector. In *Proc. ICASSP.*

Tachibana, H., Uenoyama, K., & Aihara, S. (2018). Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. In *Proc. ICASSP.*

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

Taylor, P., & Black, A. W. (1999). Speech synthesis by phonological structure matching. In *Proc. Eurospeech.*

Tiedemann, Y., Jörg & Scherrer. (2017). Neural machine translation with extended context. In *Proc. DiscoMT.*

Tokuda, K., Kobayashi, T., Masuko, T., & Imai, S. (1994). Mel-generalized cepstral analysis—a unified approach to speech spectral estimation. In *Proc. ICSLP.*

Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., & Kitamura, T. (2000). Speech parameter generation algorithms for hmm-based speech synthesis. In *Proc. ICASSP.*

Tseng, C.-y., & Su, Z.-y. (2008). Discourse prosody context-global f0 and tempo modulations. In *Proc. Interspeech.*

Tyagi, S., Nicolis, M., Rohnke, J., Drugman, T., & Lorenzo-Trueba, J. (2020). Dynamic prosody generation for speech synthesis using linguistics-driven acoustic embedding selection. In *Proc. Interspeech.*

Van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. In *Proc. SSW.*

Van den Oord, A., Vinyals, O., & Kavukcuoglu, K. (2017). Neural discrete representation learning. In *Proc. NIPS.*

Van Dijk, T. A. (1999). Context models in discourse processing. *The construction of mental representations during reading*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Proc. NIPS.*

Viswanathan, M., & Viswanathan, M. (2005). Measuring speech quality for text-to-speech systems: development and assessment of a modified mean opinion

score (MOS) scale. *Computer speech & language*, *19*(1).

Wagner, P., Beskow, J., Betz, S., Edlund, J., Gustafson, J., Eje Henter, G., ... others (2019). Speech synthesis evaluation—state-of-the-art assessment and suggestion for a novel research program. In *Proc. SSW.*

Wang, T., Yi, J., Fu, R., Tao, J., & Wen, Z. (2022). Campnet: Context-aware mask prediction for end-to-end text-based speech editing. In *Proc. ICASSP.*

Wang, Y., & Guo, M. (2014). A short analysis of discourse coherence. *Journal of Language Teaching and Research*, *vol. 5*(2).

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., ... others (2017). Tacotron: Towards end-to-end speech synthesis. In *Proc. Interspeech.*

Wang, Y., Stanton, D., Zhang, Y., Skerry-Ryan, R., Battenberg, E., Shor, J., ... Saurous, R. A. (2018). Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. In *Proc. ICML.*

Watts, O., Wu, Z., & King, S. (2015). Sentence-level control vectors for deep neural network speech synthesis. In *Proc. Interspeech.*

Wells, D., & Richmond, K. (2021). Cross-lingual transfer of phonological features for low-resource speech synthesis. In *Proc. SSW.*

Wester, M., Valentini-Botinhao, C., & Henter, G. E. (2015). Are we using enough listeners? No!—an empirically-supported critique of Interspeech 2014 TTS evaluations. In *Proc. Interspeech.*

Wiegreffe, S., & Pinter, Y. (2019). Attention is not not explanation. In *Proc. EMNLP.*

Williams, J., Rownicka, J., Oplustil-Gallegos, P., & King, S. (2020). Comparison of speech representations for automatic quality estimation in multi-speaker Text-to-Speech synthesis. In *Proc. Speaker Odyssey.*

Wolf, T., Chaumond, J., Debut, L., Sanh, V., Delangue, C., Moi, A., ... others (2020). Transformers: state-of-the-art natural language processing. In *Proc. EMNLP.*

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... others (2020). Huggingface's transformers: state-of-the-art natural language processing. In *Proc. ACL.*

Wu, N.-Q., Liu, Z.-C., & Ling, Z.-H. (2022). Discourse-level prosody modeling with a variational autoencoder for non-autoregressive expressive speech synthesis. In *Proc. ICASSP.*

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... others (2016). Google's neural machine translation system: bridging the gap between human and machine translation. *ArXiv preprint*.

Wu, Y., Wang, X., Zhang, S., He, L., Song, R., & Nie, J.-Y. (2022). Self-supervised context-aware style representation for expressive speech synthesis. In *Proc. Interspeech.*

Xu, G., Song, W., Zhang, Z., Zhang, C., He, X., & Zhou, B. (2020). Improving prosody modelling with cross-utterance bert embeddings for end-to-end speech synthesis. In *Proc. ICASSP.*

Xu, P., Saghir, H., Kang, J. S., Long, T., Bose, A. J., Cao, Y., & Cheung, J. C. K. (2019). A cross-domain transferable neural coherence model. In *Proc. ACL.*

Xue, L., Soong, F. K., Zhang, S., & Xie, L. (2022). ParaTTS: Learning Linguistic and Prosodic Cross-sentence Information in Paragraph-based TTS. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Yamazaki, Y., Chiba, Y., Nose, T., & Ito, A. (2021). Neural spoken-response generation using prosodic and linguistic context for conversational systems. In *Proc. Interspeech.*

Yan, Y., Tan, X., Li, B., Zhang, G., Qin, T., Zhao, S., ... Liu, T.-Y. (2021). Adaspeech 3: adaptive text to speech for spontaneous style. In *Proc. Interspeech.*

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). XLnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, *32*.

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

Zen, H. (2015). Acoustic modeling in statistical parametric speech synthesis-from HMM to LSTM-RNN. In *Proc. MLSLP.*

Zen, H., Agiomyrgiannakis, Y., Egberts, N., Henderson, F., & Szczepaniak, P. (2016). Fast, compact, and high quality LSTM-RNN based statistical parametric speech synthesizers for mobile devices. In *Proc. Interspeech.*

Zen, H., Dang, V., Clark, R., Zhang, Y., Weiss, R. J., Jia, Y., ... Wu, Y. (2019). LibriTTS: A corpus derived from LibriSpeech for Text-to-Speech. In *Proc. Interspeech.*

Zen, H., & Sak, H. (2015). Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *Proc. ICASSP.*

Zen, H., Senior, A., & Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. In *Proc. ICASSP.*

Zen, H., Tokuda, K., & Black, A. W. (2009). Statistical parametric speech synthesis. *Speech communication*, *51*(11).

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). *Dive into deep learning.* Open source book online.

Zhang, Y.-J., Pan, S., He, L., & Ling, Z.-H. (2019). Learning latent representations for style control and transfer in end-to-end speech synthesis. In *Proc. ICASSP.*

Zhu, X., Zhang, Y., Yang, S., Xue, L., & Xie, L. (2019). Pre-alignment guided attention for improving training efficiency and model stability in end-to-end speech synthesis. *IEEE Access*, *vol. 7*.