



UNIVERSITÄT DES SAARLANDES
Department of Language Science and Technology

MASTER'S THESIS
MSc Language Science and Technology

A Dynamic Deep Learning Approach for Intonation Modeling

Author:
Francesco Tombini
(2554549)

Supervisors:
Dr. Ingmar Steiner
Dr. Sébastien Le Maguer

Summer Semester 2018

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Saarbrücken, June 18, 2018

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Saarbrücken, June 18, 2018

Contents

List of Figures	iii
List of Tables	v
List of Acronyms	vii
Introduction	1
1 Scope	3
1.1 TTS Paradigms	3
1.2 Intonation Models	10
1.3 Proposed Methodology	16
2 Feature Extraction	19
2.1 Sampling	19
2.2 Textual Labels	25
2.3 Intonation Labels	30
3 Intonation Modeling	45
3.1 General Overview	46
3.2 Architectural Details	47
3.3 Accuracy Issues	52
3.4 Overfitting Issues	53
3.5 Data augmentation	54
4 The Segmental Synthesizer	59
4.1 General Overview	60
4.2 Description of the Input	61
4.3 Description of the Output	61
4.4 Description of the DNN Model	62
4.5 Wave Generation	62

5 Evaluation	63
5.1 Stimuli Preparation	63
5.2 Evaluation protocol	64
5.3 Results and Discussion	67
6 Conclusions	69
6.1 Perspectives and Future Work	70
Bibliography	73
A Intonation Model Label Description	83
B Lemmata Label Description	87
C Segmental Synthesizer Label Description	93
D Evaluatation Results	95
E The Implementation	97
E.1 Intonation Model Implementation	97
E.2 Segmental Synthesizer Implementation	100

List of Figures

1.1	SPSS pipeline	7
1.2	Pierrehumbert model finite state grammar	12
1.3	The Fujisaki model	13
1.4	IPO model contour approximation	14
2.1	Fixed sampling rate	21
2.2	3 anchor-point sampling	22
2.3	7 anchor-point sampling	22
2.4	Adaptive sampling	23
2.5	Example of an input vector	29
2.6	Example of a quantized contour	33
2.7	Contour translation	36
2.8	Contour dilation	37
2.9	Reconstructed contour	39
2.10	Interpolated fundamental frequency (F_0) train distribution	41
2.11	Reconstructed F_0 train distribution	42
2.12	Pitch interval train distribution	43
2.13	Example of an output vector	43
3.1	Intonation model pipeline	46
3.2	Pseudo-periodicity of long contours	56
3.3	Interpolated F_0 test distribution	57
3.4	Pitch interval test distribution	57
4.1	Segmental synthesizer pipeline	60
5.1	Evaluation questionnaire	65
5.2	Evaluation step	66
5.3	Evaluation results	67

List of Tables

1.1	Unit selection speech corpus.	5
2.1	Intonation model textual features.	28
2.2	Intonation model F_0 features	42
A.1	Intonation model label description.	85
C.1	Segmental synthesizer label description.	94
D.1	Native speakers evaluation results.	95
D.2	Non-native speakers evaluation results.	95
D.3	Native speakers evaluation results (in percentage).	95
D.4	Non-native speakers evaluation results (in percentage).	96

List of Acronyms

AM autosegmental-metrical

BAP band aperiodicity parameters

BRNN bidirectional recurrent neural network

CMU Carnegie Mellon University

CWT continuous wavelet transform

DCT discrete cosine transform

DNN deep neural network

F_0 fundamental frequency

FFNN feed-forward neural network

GMM Gaussian mixture model

GRU gated recurrent unit

HMM hidden Markov model

JND just noticeable difference

LSTM long short-term memory

MCD mel-cepstral distortion

MGC mel-generalized cepstrum

ML machine learning

MOS mean opinion score

MT machine translation

MTL multi-task learning

MUSHRA multiple stimuli with hidden reference and anchor

NAG Nesterov accelerated gradient

OALD Oxford Advanced Learner's Dictionary

POS part-of-speech

RMSE root mean squared error

RNN recurrent neural network

SGD stochastic gradient descent

SNN self-normalizing neural network

SPSS statistical parametric speech synthesis

SPTK Speech Signal Processing Toolkit

ToBI tones and break indices

TTS text-to-speech

USS unit selection synthesis

VUV voiced-unvoiced

Introduction

Over the last decade, text-to-speech (TTS) technology has advanced by leaps and bounds. This is especially true for statistical parametric speech synthesis (SPSS), one the most widely used TTS paradigms. The main reason for its popularity is its extreme flexibility, which makes it particularly attractive for research.

Although SPSS has reached high levels of intelligibility, one thorny and long-standing issue is the modeling of prosody. Prosody is a complex speech phenomenon that pertains to the suprasegmental properties of speech, i.e., properties beyond single sounds. Even though there is no agreed-upon set of prosodic variables, the classical acoustic parameters describing prosody are: F_0 (responsible for intonation), duration (responsible for absolute and relative duration of units), intensity (responsible for energy-related features such as stress) and spectral characteristics (responsible for voice quality).

As systems are getting better at modeling segmental features, their inadequacies in modeling suprasegmental features are becoming increasingly apparent. This is especially true with regard to intonation. Even though current approaches can produce fairly natural and intelligible speech, their intonation tends to be rather bland and monotonous, which is unsuitable for many TTS applications.

Despite the crucial role that intonation plays in making synthetic speech sound more natural and human-like, for the longest time the modeling of intonation has largely been subordinated to that of segmental features. This is perhaps due, on the one hand, to the higher salience of segmental features and, on the other, to the more elusive and generally intractable nature of intonation.

The reason why modeling intonation is such a hard task is due to the fact that it carries out a very large number of complex and highly-entangled linguistic, para-linguistic and extra-linguistic functions, whose relationship to the source text is anything but trivial.

Nevertheless, intonation remains a rather pressing research topic, that

is bound to become more critical in the coming years, especially as the demand for TTS applications such as dialog systems, virtual agents, and personal assistants increases. For these applications, it is desirable to generate speech that is not just intelligible, but also affective and natural enough that users forget they are interacting with a computer. In these domains, intonation is still very much considered an open question.

For these reasons, in my thesis I will be focusing on the modeling of intonation. Within the scope of this proposal, intonation modeling is used to refer to the problem of generating a sequence of F_0 values given a text sequence. Here, other important prosodic phenomena that accompany intonation such as duration, loudness, timbre, etc., will be ignored or assumed to be known.

This proposal is structured as follows. In Chapter 1, I will offer a brief overview of TTS technology and intonation models. In Chapter 2, Chapter 3 and Chapter 4, I will propose, motivate, and justify a deep learning methodology for the modeling of intonation in the context of SPSS. In Chapter 2, I will discuss how feature extraction is carried out in the proposed methodology. In Chapter 3, I will present a dedicated deep neural network (DNN) model for the modeling of intonation. In Chapter 4, I will present a DNN model for the synthesis of segmental features from a linguistic specification and an F_0 contour. In Chapter 5, the proposed methodology is evaluated and compared to a state-of-the-art parametric TTS system. Finally, in the last chapter I will offer my own conclusions and perspectives for future research.

Chapter 1

Scope

This chapter is devoted to a brief overview of the state-of-the-art in TTS technology and it will serve as a way of contextualizing the scope of my work within the field.

In the first section of the chapter, I will centre the discussion around the dominant TTS paradigms and the role that F_0 modeling plays within them. This serves as justification for the TTS paradigm that was selected for the implementation of the proposed methodology.

The second section will be devoted to the discussion of models of intonation. In this section, I will offer a brief review of some of the historically most influential models of intonation.

In the third section, I will outline a deep learning methodology for the modeling of intonation. The methodology is contextualized and compared with previous research.

1.1 TTS Paradigms

TTS technology can be divided into two main branches. On the one hand, we have rule-based synthesis and on the other we have corpus-based synthesis.

In rule-based synthesis, which is historically highly related to formant-synthesis, expert knowledge is required to come up with rules that adequately account for speech phenomena. This knowledge is expensive to acquire, it does not apply cross-linguistically, and it does not produce very natural sounding speech, as it is almost impossible for experts to describe any meaningful size of the combinatory space by means of rules.

Most rule-based intonation systems are inspired by the Pierrehumbert model and make use of some variation of the tones and break indices (ToBI)

transcription system (Silverman et al., 1992).

The main advantage of rule-based approaches is that it can be very flexible and it can have a very small footprint. Even though this technology has largely been surpassed by corpus-based synthesis, rule-based synthesis is still been used for some limited applications, where footprint might be a concern or when no speech data is available.

The second major approach to TTS is corpus-based synthesis. Within this approach, we do not try to come up with linguistic rules that describe speech phenomena, but rather we try to build models or databases from speech corpora.

As this approach is currently regarded as the state-of-the-art, the following sections will be devoted to two main paradigms of corpus-based synthesis: unit selection synthesis (USS) and SPSS.

1.1.1 Unit Selection Synthesis

USS is based on the use of original speech signals found in speech corpora, where units of speech are dynamically selected from the corpus at run-time. The selection process, in this case, is guided by the concerted effort of minimizing target costs (i.e., how close the candidate is to our target specification) and concatenative costs (i.e., how well the candidate fits with adjacent units).

Many of the foundational ideas behind this approach were first laid out in Hunt and Black (1996) and Campbell and Black (1997), which quickly lead to the development of systems based on this new approach. The success of this approach is due to the fact that, despite its simplicity, it can produce highly intelligible and highly natural synthetic speech.

However, USS is not very flexible with regard to intonation modeling. This is because most USS systems adopt a so-called “as-is” approach, whereby the problem of prosody is tackled only indirectly by including prosodic information in the target and concatenative cost functions. This is under the assumption and the hope that the system will happen to find appropriate segments that also fit together.

However, because of the extremely high levels of sparsity that characterize most language and speech phenomena (van Santen, 1997; Möbius, 2003), USS approaches will often lead to noticeable prosodic discontinuities when no appropriate units can be found. This problem is alleviated in most USS systems by designing the corpus to include fairly neutral news-style speech, which can be problematic if one wishes to generate expressive and affective speech.

These issues relating to F_0 modeling constraints can be better illustrated by the example in Table 1.1, where we can imagine how a unit selection system would operate in order to predict the F_0 contour of the target sentence, provided that sentences 1, 2, 3, and 4 are our entire corpus.

The system would completely ignore sentence 1 and splice together the boldface segments from sentence 2, 3, and 4. This is due to the fact that the target sentence and sentence 1 have very little in common as far as phone sequence is concerned.

Target	Earlier, I drank a small glass of fizzy coke.
Corpus	<p>(1) Clearly, she has a real knack for funny jokes.</p> <p>(2) I met your husband earlier.</p> <p>(3) Did you say she drank a small glass of water?</p> <p>(4) A nice bottle of fizzy coke is what is needed on a hot summer day!</p>

Table 1.1: Example of a speech corpus, where the second row contains the speech corpus used by a system for the synthesis of the target utterance contained in the first row.

As we can see, intonation of the boldface segments would not match what is required for the target sentence.

For instance, “earlier” in sentence 2 would have a far flatter and descending F_0 compared to the target, because that is what we expect at the end of a declarative sentence. The segment “drank a small glass” comes from a question and the F_0 will be rising too much for our needs. Finally, “of fizzy coke” is probably going to have a really upbeat and lively intonation because it’s contained in an exclamatory utterance.

However, we can easily make an argument that the intonation from sentence 1 would be a far more suitable candidate than the intonation generated from the combination of the boldface segments. Unfortunately, USS systems are unable to make use of prosodically more appropriate segments as these will rarely satisfy the phone-driven target specification. This entails that even though the corpus might be rich in useful prosodic information, it will remain largely unused.

1.1.2 Statistical Parametric Speech Synthesis

After almost over a decade dominated by USS, where the task of speech synthesis was essentially reduced to a highly sophisticated talking clock,

we have recently been witnessing to a resurgence of older approaches, but revisited in the light of new ideas and technology. One such example is SPSS.

In SPSS, we do not concatenate or manipulate the original pre-recorded waveforms, but rather, signals are synthesized from scratch, similarly to what used to be done in formant synthesis.

The main innovation of SPSS lies in reframing the problem as a machine learning (ML) task. Unlike formant synthesis, we no longer try to come up with expert-knowledge rules, but we use ML algorithms to automatically infer statistical models that describe mappings between sets of input features and output features.

The main difference with USS is that, in SPSS, we do not use data directly in the output, as we only use the corpus to extract linguistic and acoustic features that can later be used to train models. In USS, we can never do away with the original data, whereas in SPSS, once we train a model, we no longer need the original waveforms to synthesize new utterances. In USS, we do not have a model proper, but something more akin to a database, as none of the original data has been parametrized.

The reason why research on alternatives to USS such as SPSS and articulatory synthesis is so active is because of the flexibility these alternative approaches can offer. Unlike USS, where our control is limited to how snippets of the pre-recorded waveform can be arranged, in parametric approaches, we theoretically have absolute control over every aspect of speech production, including duration, pitch, spectral features, etc. This freedom comes from the fact that we are no longer dealing with cumbersome data that we cannot easily manipulate without significant signal degradation. As signals are synthesized from scratch, we can define and set the parameters that are necessary to generate the speech phenomena that we want to capture.

This is especially important with regard to the F_0 , an aspect that has been particularly hard to control under the unit selection paradigm, with the consequence that intonation modeling for a long time was mostly subordinated to the modeling of segmental features. In SPSS, it is possible to build models of intonation independently of other aspects of speech. This means we have far more freedom to experiment with different approaches without worrying too much about other aspects of speech production.

Although the classical way of describing TTS systems is in terms of front-end and back-end, my overview will be structured in terms of an online and offline stage. As the front-end is usually quite similar for most TTS systems, it is more useful to structure the overview in terms of a more

general ML approach, where we often distinguish between an offline stage (i.e., training) and an online stage (i.e., inference). Figure 1.1 shows the pipeline of a typical SPSS system.

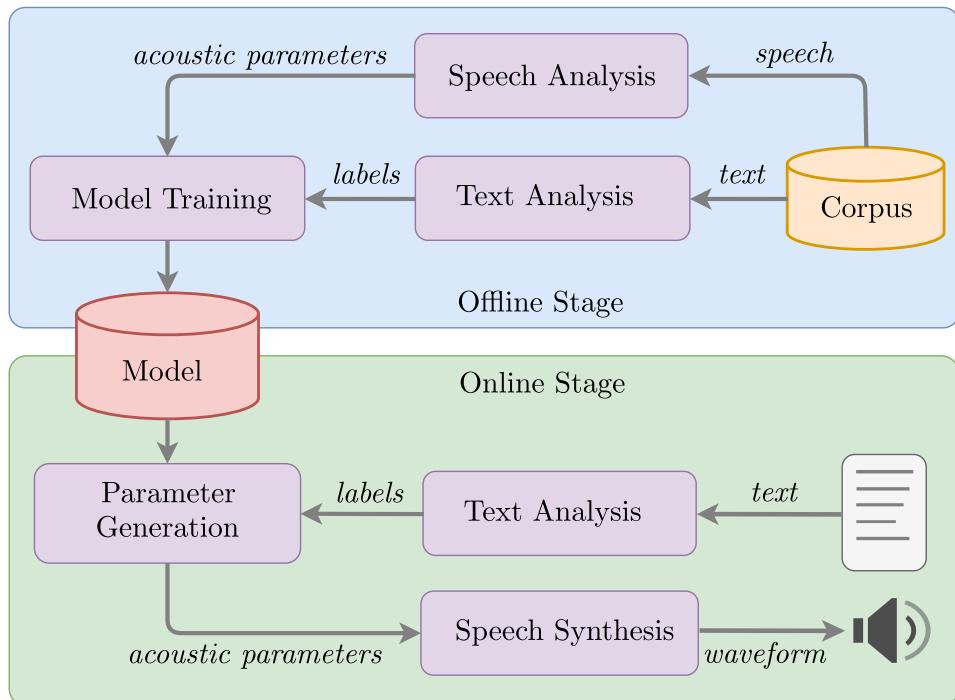


Figure 1.1: Pipeline of a typical SPSS system. The blue box represents the offline stage, whereas the green one the online one.

The Offline Stage

In the offline stage (the blue box in Figure 1.1), we first compile a speech corpus, i.e., a collection of audio material and its corresponding text, which is going to function as the training data.

From the corpus, we extract speech and textual features. The input and output features must be aligned along the time domain. This can be achieved by means of manual corpus annotation, or by means of forced alignment.

A text analysis component extracts labels that are used to construct a linguistic specification. The linguistic specification usually contains information about phone labels, stress, part-of-speech (POS) tags, etc.

A speech analysis component extracts output features in the form of acoustic parameters that can be used by a wave synthesizer to generate a waveform. The exact nature of the acoustic parameters will largely depend on the selected wave synthesis mechanism.

The most common approach is to utilize a vocoder, which can decompose waveforms into acoustic parameters. Many commonly used vocoders are based on the source-filter theory of speech, which arose from the merging of various theories on speech and acoustics (Chiba and Kajiyama, 1941; Fant, 1960). Two very commonly used vocoders are STRAIGHT (Kawahara et al., 1999) and WORLD (Morise et al., 2016). These vocoders can produce reasonably intelligible speech, but they are also infamously known for the buzzing quality of the synthetic signal.

There exists a different class of vocoders, which completely ignore the source-filter model of speech and instead are based on sinusoidal modeling mechanisms (McAulay and Quatieri, 1986). An open-source example of these vocoders proposed recently is the MagPhase vocoder (Espin et al., 2017). These vocoders produce much better speech quality, but they are harder to use in real-world speech applications.

The labels produced by the text analysis component and the acoustic parameters extracted by the speech analysis component are then used to train a model that learns mappings from input to output features. hidden Markov models (HMMs) and DNNs are the two most common approaches for the training of the model.

The first SPSS systems were based on HMM approaches. Even though they are quite effective, they are also plagued by a number of limitations. As pointed out in Zen et al. (2013), decision trees and Gaussian mixture models (GMMs), which are a big part of HMM approaches, are not particularly suited for the modeling of complex long-term dependencies, causing the speech to be less varied and fairly monotonous. Decision trees are also

not particularly suited to approximating complex functions, at least not without building prohibitively large models. Finally, decision trees tend to partition the input space into smaller regions, which results in a very fragmented representation of the input space. This is very inefficient, as it requires the use of separate sets of parameters for each region. It has been shown that this can lead decision trees to completely ignore weak or rare input signals, such as word-emphasis (Yu et al., 2010).

All of these issues, coupled with the recent emergence of deep-learning technology, have propelled the almost exclusive adoption of neural networks in almost every domain of speech technology. Most neural networks used nowadays in deep learning are sets of algorithms based on mathematical models loosely inspired by biological neurons. The power of neural networks rests on their ability to function as universal approximators, which endows them with the capacity to recognize and model extremely complex patterns.

DNNs solve many of the shortcomings affecting HMM-based systems. DNNs can model very complicated functions, they can make use of weak or rare input signals, they can deal with very high-dimensional and highly correlated features and they produce very compact and distributed representations, which means model parameters are used much more efficiently. These properties are all highly desirable when solving very complicated problems such as speech and language tasks, where it is not at all uncommon to have to deal with very sparse and high-dimensional input spaces and where features interact in a very subtle and hierarchically layered fashion.

Over the last few years, a large body of research has emerged in which researchers have attempted to make use of DNNs to model speech data. First attempts at using DNNs in SPSS involved only replacing the GMM components of a HMM system (Ling et al., 2013) with feed-forward neural networks (FFNNs).

As time went on and as DNNs have consistently been shown to outperform HMM-based systems with similar numbers of parameters (Zen et al., 2013), more and more components in SPSS systems have been replaced: from the prediction of spectral features, to F_0 modeling, to duration modeling, etc.

At the same time, we also witnessed the introduction of increasingly powerful and sophisticated neural network architectures such as recurrent neural networks (RNNs). RNNs made it possible to overcome the limitations imposed by FFNNs as far as the modeling of contextual information is concerned. Although, theoretically FFNNs can be used in the context

of time-series prediction, the contextual information is limited by the size of the input layer. Recurrent structures solve this problem by introducing recurrent layers that are reused at each time step of the graph. This allows us to feed the network arbitrarily long inputs.

RNNs became especially popular once the infamous *vanishing gradient* (Bengio et al., 1994) problem had been solved by introducing new flavors of RNNs such as long short-term memories (LSTMs) (Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRUs) (Chung et al., 2014). These architectures made it possible to train RNNs on very long sequences of data without vanishing gradients by introducing cells with input, output, and forget gates that better regulate the flow of information across various time steps.

The aforementioned trend whereby more and more components of the SPSS pipeline are removed in favor of a more end-to-end approach has been pushed to its limits in the last few years thanks to the introduction of architectures such as Char2Wav (Sotelo et al., 2017), Deep Speech (Hannun et al., 2014) and Tacotron (Wang et al., 2017b).

The Online Stage

Once a model has been trained and tuned, it can be brought online and used to synthesize novel speech. This constitutes the online stage of the SPSS pipeline shown as the green box in Figure 1.1.

At this stage, the end-user can provide some text to be converted into speech. Similarly to what happens in the offline stage, the text undergoes text analysis to extract textual labels to build a linguistic specification. The linguistic specification is used to query the model to generate acoustic parameters. Finally, a speech synthesis component converts the acoustic parameters into a waveform.

1.2 Intonation Models

In this section, I will offer a brief overview of the historically most influential theories and models of speech intonation that have successfully been implemented in the context of TTS. Special attention will be given to the representational level of the F_0 , as well as the main underlying assumptions posited by each model.

In keeping with what is done in many publications on intonation modeling, neural network approaches are covered alongside the classical intonation models previously discussed, even though, in my estimation, HMMs

and DNNs do not constitute models of intonation proper.

It is important to separate two fundamental levels involved in the modeling of F_0 . On the one hand, we have a representational level of the F_0 . This aspect is primarily concerned with the description and formulation of F_0 representations that are based on very clear and predefined theoretical assumptions about the nature of intonation. Some of these assumptions are rooted in the field of phonology (i.e., the Pierrehumbert model), phonetics (i.e., the Fujisaki model), or perception (i.e., the IPO model), etc. In a ML perspective, this aspect can be viewed as a feature engineering step, in which we inject some of our assumptions about the phenomenon we want to model into the representation or encoding of the F_0 .

A second level that we need to distinguish from the first one is the mapping between the input features to the F_0 representation, which constitutes its own separate model built on top of the intonation model. The mapping could be codified by finite state grammars (e.g., most implementations of the Pierrehumbert model) or by statistical models such as HMMs or DNNs. Although the modeling of the mappings between inputs and outputs depends greatly on the intonation model that was used to produce the F_0 representations, the two should not be conflated (e.g., there exist implementations of the Fujisaki model both with HMMs (Yoshizato et al., 2012) or DNNs (Sakurai et al., 2000)).

1.2.1 Pierrehumberts Theory of Intonation

The Pierrehumbert model of intonation, first introduced by Pierrehumbert (1980), can be regarded as one of the most influential exponents of the tone sequence school. Largely based on autosegmental-metrical (AM) phonology theory, the tone sequence school views contours as linear sequences of independent tones (or pitch accents).

Crucially, the tones within the sequence do not interact with each other, but rather they act as a sequence of units, where their contrastive features give rise to distinct intonational meanings.

The Pierrehumbert model uses two fundamental tones as its main building blocks: a high (H) tone and low (L) tone. These tones are combined to give rise to three larger tone units: pitch accents, phrase tones and boundary tones.

Pitch accents, which describe prosody at the word level, are marked by a “*” and they are made up of either a single tone (H*, L*) or two tones (H*+L, H+L*, L*+H, L+H*), where the position of the “*” indicates the position of the stressed syllable.

Multiple pitch accents can be combined into larger prosodic phrases that describe the prosody of intermediary phrases. The tones used in the context of intermediary phrases are phrase tones, and they are marked by a “–” (H–, L–).

Finally, intermediary phrases can be combined into intonation phrases, the largest prosodic unit accounted for by the model. Intonation phrases are marked by boundary tones placed at the edges of the intonation phrase. Boundary tones are marked by a “%” (H%, L%, %H, %L), where the position of the “%” depends on placement of the tone either at the onset or at the end of the intonation phrase.

To make sure only well-formed sequences can be produced, the Pierrehumbert model also specifies all the possible combinations of tones by means of a finite state grammar.

$$\left(\left\{ \begin{array}{l} \%H \\ \%L \end{array} \right\} \left(\left\{ \begin{array}{l} H^* \\ L^* \\ H^*+L \\ H+L^* \\ L^*+H \\ L+H^* \end{array} \right\}^+ \left\{ \begin{array}{l} H- \\ L- \end{array} \right\} \right)^+ \left\{ \begin{array}{l} H\% \\ L\% \end{array} \right\} \right)^+$$

Figure 1.2: Finite state grammar of the Pierrehumbert model.

The Pierrehumbert model has led to the creation of the widely used transcription system ToBI. Even though the ToBI system was originally designed for American English, it has since been adapted to many languages and implemented as part of many TTS systems.

1.2.2 The Fujisaki Model

The Fujisaki model of intonation, first introduced by Fujisaki (1983), can be regarded as one of the most influential exponents of the superimposition school. Unlike the Pierrehumbert model, the F_0 is not viewed as a sequence of tones, but rather as a complicated function generated by the superimposition of smaller components.

The Fujisaki model posits the existence of two main components that determine the shape of the F_0 contour: the phrase command and the accent command.

The phrase command is responsible for the modeling of the global intonation of the utterance, whereas the accent command controls local pitch excursions (for example in stressed syllables or stressed words).

Phrase commands are modeled by pulses, and accent commands, by step functions. The outputs of these operations are then joined by means of an additive approach.

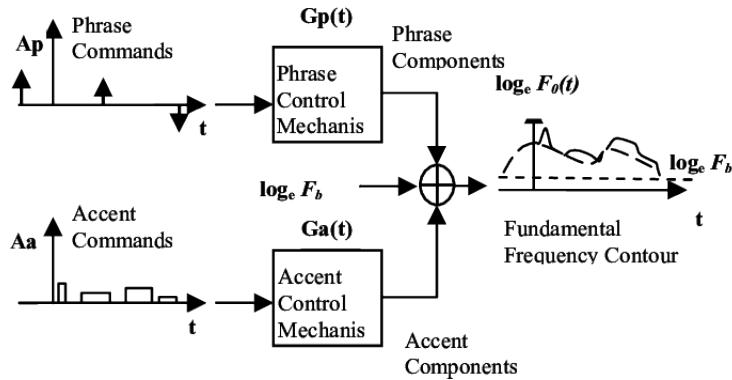


Figure 1.3: The Fujisaki Model. Reprinted from Fujisaki (2002).

Phrase commands and accent commands are said to simulate the action of laryngeal muscles that control the frequency of vibration of the vocal cords. This endows the model with a strong physiological basis.

As the placement of phrase commands and accent commands often corresponds to key phonological units such as accent groups and phrase boundaries, the model is able to produce linguistically interpretable commands.

1.2.3 The IPO Model

Initially introduced to model Dutch intonation, the IPO model of intonation (Gussenhoven et al., 1992) is generally classified as a perception-based model of intonation. This is because this approach relies on a number of assumptions about how contours are perceived by humans.

In particular, the IPO model puts forward that F_0 changes that cannot be perceived by humans need not be modeled and that it is sufficient to only model perceptually salient features of pitch. Additionally, the IPO approach is based on the assumption that the human ear is more sensitive to tone variations (i.e., rise versus fall), than tone intensities (i.e., high versus low).

Because on these assumptions, the IPO approach does not model raw F_0 contours, but rather piece-wise linear approximations superimposed on a general declination line that are perceptually indistinguishable from the original, as shown in Figure 1.4.

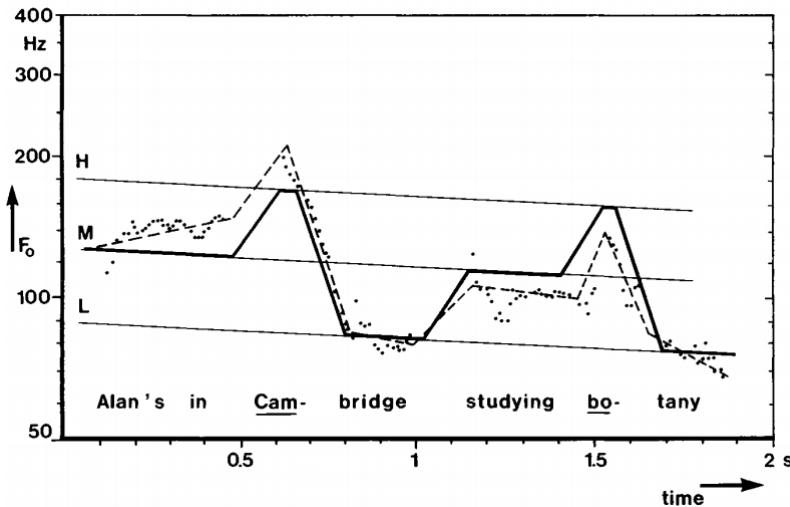


Figure 1.4: Example of a piece-wise linear approximation of a contour. Reprinted from Gussenhoven et al. (1992, p. 49).

After these approximations are produced, contours can be broken down into a fixed inventory of F_0 movements that are capable of describing the whole original speech corpus from which they were extracted. The combinations of these F_0 movements are described by a grammar.

1.2.4 The Tilt Model

The tilt intonation model was first introduced by Taylor (1994) and is based on his *rise/fall/connection* model of intonation. As this model is based on a shape-based description of prosodic events, it has been classified as an acoustic stylization-based model.

The model is based on four building blocks: pitch accents (modeled by combinations of quadratic functions), boundary tones (also modeled by combinations of quadratic functions), connections (modeled by straight-line interpolations) and silences.

These building blocks can be used to describe prosodic events. In the model, each event is associated with a set of parameters. Each set of parameters includes the position of the intonational event within both the time and frequency domains, the amplitude, the duration, as well as a so-called *tilt* value.

A tilt value is a real-valued parameter that describes the shape of the event and is calculated by dividing the difference of the rise and fall of the amplitudes of the quadratic functions by their sum:

$$tilt = \frac{|A_{rise}| - |A_{fall}|}{|A_{rise}| + |A_{fall}|} \quad (1.1)$$

where:

A_{rise} = amplitude of the rise
 A_{fall} = amplitude of the fall

The tilt parameters can range between -1 and 1 , where -1 represents a pure fall, 0 a perfectly symmetrical peak, and 1 a pure rise.

1.2.5 HMM and DNN Approaches

Over the past couple of decades, there have been a number of attempts of modeling the F_0 by means of HMMs and DNNs. Multiple F_0 prediction models have been developed for the HMM paradigm (Latorre and Akamine, 2008; Yu et al., 2009; Yu and Young, 2011). However, as DNNs have been shown to outperform HMMs, various flavors and combinations of neural network architectures for the modeling of the F_0 have been proposed, from FFNNs (Vainio, 2001), to RNNs (Zen and Sak, 2015), to bidirectional recurrent neural networks (BRNNs) (Fan et al., 2014), etc.

Aside from the type of architecture, neural network approaches can be classified based on whether prosody is modeled by a dedicated model (e.g., Vainio, 2001; Traber, 1990) or jointly with other speech features (e.g., Zen and Sak, 2015; Fan et al., 2014). The latter approach has become more common in the last few years and is consistent with the more general trend in TTS technology towards increasingly end-to-end systems. This is also the approach found in the open-source Merlin toolkit for DNN synthesis (Wu et al., 2016).

The main idea behind this trend is to minimize human effort and intervention in the task of feature engineering as much as possible. In spite of its convenience, this approach is also less modular and less flexible, especially for the purpose of research.

DNNs are notoriously hard to examine, because they produce highly distributed representations, where it is not easy to determine for which aspects each parameter is responsible. When DNN F_0 modeling is performed jointly with other speech features, the F_0 contour is intrinsically tied to the other features. This means that if we try to replace the F_0 contour with a different one, the other features will not change to suite the new contour.

Another aspect by which neural network approaches may differ is how much (or how little) the F_0 contour is pre-processed. One extreme case is

when F_0 contours are only minimally processed by means of a simple log transform. Aside from this extreme, there are a number of approaches with various levels of pre-processing such as Fujisaki model inspired approaches (Sakurai et al., 2000), F_0 template approaches (Ronanki et al., 2016), quantization approaches (Wang et al., 2017a), discrete cosine transform (DCT) approaches (Yin et al., 2016), etc.

Among the previous work focusing on intonation modeling in SPSS, where significant processing is carried out, the most recent studies concern the use of wavelets (Suni et al., 2013; Vainio et al., 2013; Ribeiro and Clark, 2015; Ribeiro et al., 2016). In particular, the continuous wavelet transform (CWT) has proven useful both for analysis (Vainio et al., 2013), as well as modeling (Suni et al., 2013). Further, it has been shown that CWT can be used to decompose the F_0 contour into various components based on different prosodic scales (phrase, word, syllable, etc.), where not all components are equally relevant for the reconstructed signal (Ribeiro and Clark, 2015). Low frequencies do not convey much prosodic information, and high frequencies are mostly noise. Wavelet contours can be stylized for each prosodic level by means of DCT. This approach was successfully applied also in the context of DNN synthesis, and it was shown to produce more natural sounding prosody (Ribeiro et al., 2016).

1.3 Proposed Methodology

In this section, I outline a deep learning methodology for the modeling of intonation in the context of SPSS. The advantages of SPSS have been amply explored in Section 1.1, including the freedom to model the F_0 independently of other speech features. In the proposed methodology, the F_0 is generated by a dedicated DNN model, while other acoustic features are generated by a separate DNN model implemented for the synthesis of segmental features.

In Section 1.2, I presented the historically most influential theories, models and approaches for the modeling of intonation. One common feature shared by most of the previous work is either a fully or partially static representation of intonation. For instance, Pierrehumbert’s theory of intonation is based on a binary contrast of static high and low tones, the Fujisaki model is based on the superimposition of functions that output static contour values, wavelets decompose contours into static hierarchically organized sub-components, etc.

The IPO and the Tilt models display a higher level of dynamism as contours are described in terms of fall and rise. However, these approaches

are not fully dynamic. For instance, under the IPO approach, contours are synthesized from a fixed inventory of standardized pitch templates, where the position of the templates within the frequency domain is determined by the general declination line on which they are superimposed.

The Tilt model is more flexible than the IPO model, as the shape of prosodic events is not determined by a fixed inventory of pitch movements, but rather by a set of *tilt* parameters. The Tilt model is also somewhat dynamic, because each event is described in terms of fall/rise, instead of a static high/low opposition. However, the start of each event is a static frequency value that anchors the fall/rise pattern within a specific region of the frequency domain.

In this thesis, I propose a purely dynamic approach, in which we model the dynamic evolution of the interpolated F_0 through time from a starting position. The dynamic information is parametrized by a sign value for the direction of change, and a quantized magnitude value for the amount of change in such direction. The overall position of the predicted contour within the frequency domain is determined based on the speaker's voice register.

Unlike the Fujisaki model and wavelet approaches, in the proposed methodology, contours are not decomposed into hierarchical sub-components. Similarly to the Tilt model, they are encoded into sequences of intonational events, where each event represents a pitch movement. Unlike the Tilt model, the position of each event within the frequency domain is not static, but rather relative to the previous event.

Finally, the proposed methodology takes advantage of the recent advances in deep learning technology by also including word embeddings in the model, so that contour predictions are driven by semantically richer information.

Chapter 2

Feature Extraction

In this section, I will describe how feature extraction is carried out in the proposed methodology.

This section is divided into three parts. In the first part, I will discuss the two most commonly used approaches for sampling speech data, and I will present the adaptive sampling method implemented for my methodology. In the second part, I will provide a description of the textual information that I decided to include and justification for each chosen linguistic feature. In the third part, I will present the technique I used for the dynamic encoding of the interpolated F_0 .

2.1 Sampling

One major preliminary step in approaching the problem of F_0 modeling is to decide how the pitch information and the corresponding linguistic labels should be sampled. Most F_0 estimation tools produce as their output a sequence of F_0 values sampled at a constant rate. Typically, for most TTS systems, acoustic information is sampled at a 5 ms time interval. However, different researchers might decide to use this data in very different ways depending on how they approach the problem.

In this section, I will present the sampling approach used in my implementation. The proposed sampling scheme is based on the selection of a support level and a default time interval. The size of the interval is adapted based on the duration of the support level. This approach is a hybrid of two common sampling approaches: frame-by-frame and anchor-point methods. The advantages and disadvantages of these methods are discussed and used as motivation for the adaptive sampling rate method presented here.

2.1.1 Frame-by-Frame Approach

Under this approach, the data is sampled at a fixed time interval, typically at a 5 ms time interval. This is what we typically find in TTS systems such as Merlin (Wu et al., 2016), where the F_0 and other acoustic features are modeled jointly in a frame-by-frame fashion.

The main advantage of frame-by-frame approaches is that they are very convenient and they make the least assumptions about the data. However, it is also needlessly wasteful, as we would have to make a prediction for every single frame. This can be problematic, if we wish to include very rich input features such as word embeddings. Under this approach, using word embeddings with hundreds (if not thousands) of inputs per frame could become prohibitively expensive.

As a lot of the information contained in the F_0 contour is redundant, most of it can be discarded, as it is easily reconstructed from partial or parametrized contours by means of interpolation. Not only does this allow for larger inputs, but having fewer points to predict also means faster and more efficient training, especially for longer sequences.

Despite the recent introduction of RNNs, which allow for much better modeling of long sequences, long distance dependencies still remain a tricky aspect to model, because of the limited time memory that these architectures offers. Reducing the number of intonational events that we want to predict per utterance will make the learning process much easier.

However, we cannot simply compress the data by naively downsampling the output produced by F_0 estimation tools. The main problem with a lower fixed sampling rate is that the sampled points would be located in linguistically irrelevant positions. For instance some points might be located very close to the syllable boundaries, sometimes far from them. Sometimes certain syllables might be skipped entirely because they are too short. Notice for instance in Figure 2.1, how a fixed sampling rate causes the sampled F_0 to be distributed erratically with respect to linguistic segments.

2.1.2 Fixed Anchor-Point Approaches

A commonly adopted solution to the problems that arises from the adoption of low fixed sampling rates is to use so-called “anchor points”, i.e., linguistically relevant positions within the utterance (e.g., van Santen and Möbius, 1997). One might, for example, choose to sample F_0 measurements at the start, center, and end of each syllable.

The advantage of this approach is that the contour can be represented

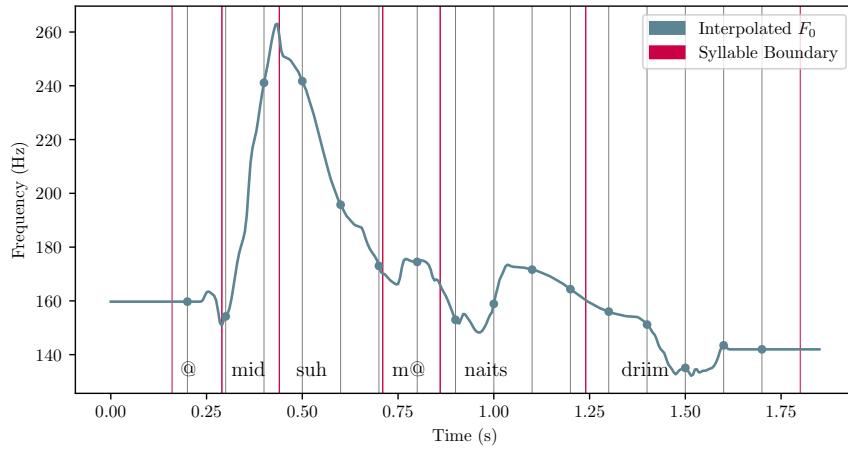


Figure 2.1: Location of extraction points in an F_0 contour at a fixed 10 Hz sampling rate.

into a much more compact format, which makes training a lot easier. Under this approach we also make sure that we have some information about each and every instance of the chosen support level (usually the syllable).

Unlike the frame-by-frame approach, this approach has to make a number of assumptions about pitch and its perception. The first assumption that we have to make is that phenomena that are lost as a consequence of the downsampling process are not that important and that reconstructed contours are still accepted by humans as natural. Additionally, we have to assume that the timing of prosodic events corresponds to linguistic units such as syllables or syllable sub-units, which might not always be the case for all natural languages. An additional assumption that we have to make if we simply use the same number of anchor points per linguistic segments is that duration has no bearing on the shape of a pitch template (unless of course duration is explicitly provided as an input feature to the model). Under this assumption, we might fail to account for physiological constraints about the vocal tract, such as how fast pitch rises or falls can be produced.

One disadvantage illustrated by using the same number of anchor points per syllable is illustrated by Figure 2.2. As we can see, with a vanilla implementation of the anchor-point approach we are unable to use more data points for longer syllables where we might observe more complex patterns.

By the same token, if we want to increase the number of anchor points

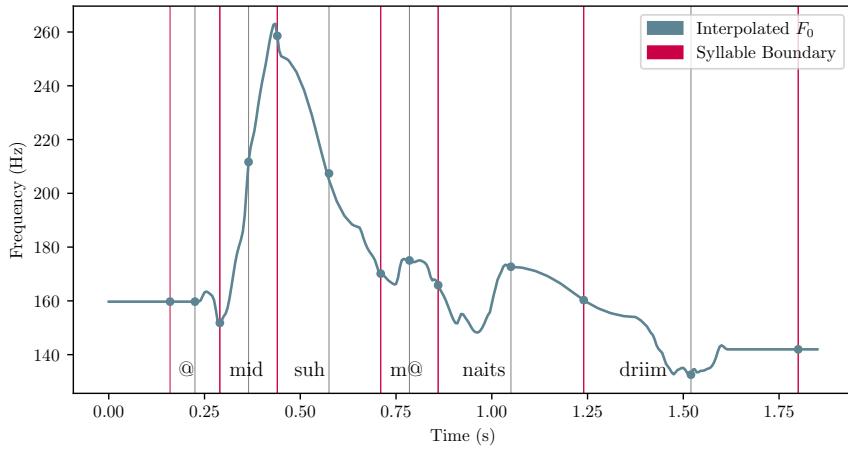


Figure 2.2: Location of extraction points in an F_0 contour with 3 anchor points.

to capture more complex behaviors in longer syllables, as it was done in Figure 2.3, we end up with too many points in shorter syllables, where contours are less complex.

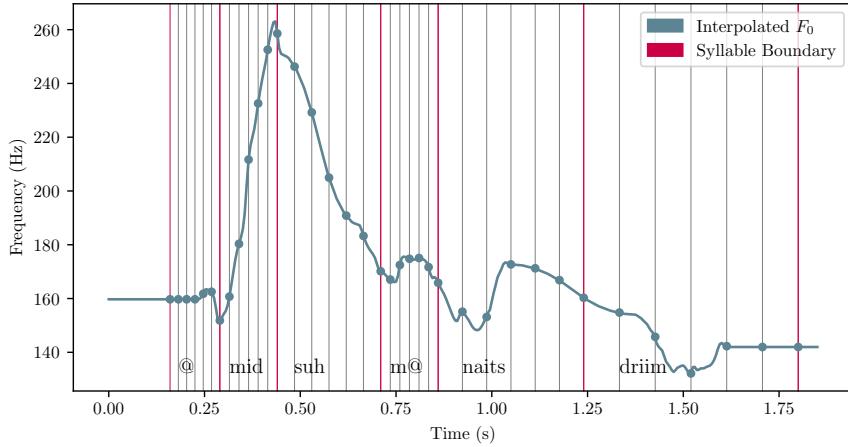


Figure 2.3: Location of extraction points in an F_0 contour with 7 anchor points.

2.1.3 Adaptive Sampling Rate

As previously mentioned, frame-by-frame sampling rates have the advantage of making fewer assumptions about the data and are more conveniently implemented. However, including rich inputs such as word embeddings is prohibitively expensive. On the other hand, anchor points can provide us with a more consistent number of points across linguistic segments, but we cannot allocate more anchor points for longer segments.

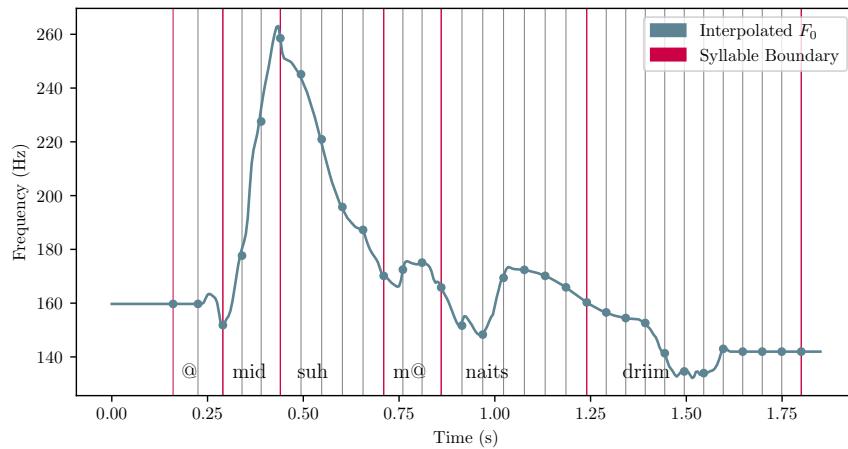


Figure 2.4: Location of extraction points in an F_0 contour with an adaptive sampling rate.

Given the various advantages and disadvantages of both approaches, in my methodology, I decided to adopt a hybrid of both. This is implemented by first selecting an appropriate linguistic level for interval subdivision (e.g., the syllable level) and a time interval as the default sampling rate (e.g., 0.1 s). The default sampling rate was determined heuristically, by trying out many different values and plotting the the original contours overlaid with vertical lines representing the sampling sites. For each syllable, the sampling rate is adapted, based on the default sampling rate and the duration of the syllable, based on Equation 2.1.

$$sr_s = \frac{dur_s}{\lceil \frac{dur_s}{dsr} \rceil} \quad (2.1)$$

where:

- sr_s = sampling rate for syllable s
- dur_s = duration of syllable s
- dsr = default sampling rate

As we can observe in Figure 2.4, the proposed adaptive sampling rate produces points of varying distance across syllables. However, less so than a purely anchor-point-driven approach. At the same time, we are able to allocate more sample points to longer syllables, which is the main advantage of the other approach, since we are not limited by a fixed number of anchor points.

2.2 Textual Labels

Once the time locations where inputs and outputs are sampled is determined, one must choose suitable input labels for the prediction of the target outputs.

In this section, I present each set of linguistic labels that are included in the proposed methodology and describe their role and usefulness for the task of F_0 modeling. Finally, I present how they are converted into their corresponding binary features and combined into input vectors for the DNN model.

2.2.1 Phones

Even at the lowest level, i.e., at the level of the phonemes, F_0 is affected. At this level, depending on the phoneme and the amount of voicing applied to its corresponding phonemic realization, pitch features might or might not be specified.

This inconvenient fact is often circumvented by interpolating the missing F_0 measurements and applying smoothing. Critics of this approach point out that this produces artifacts that might negatively affect the learning process and the results (Wang et al., 2017a).

In the proposed methodology, each intonational event represents a movement describing the amount of pitch change from the previously observed F_0 value (for more details on the F_0 encoding, see Section 2.3). If the frequency of the previous value is unspecified (because it has not been interpolated), then it is impossible to calculate the amount of relative change between it and the currently observed F_0 value. This means that in the proposed methodology, it is not possible to do away with interpolation. As F_0 must always be interpolated, it is not necessary to include information about phonemes to account for unvoiced parts of the contour. For these reasons, information about phones is not included.

2.2.2 Onset and Rhyme

In addition to the issue relating to whether phonemes are voiced or not, we also need to take into account that the timing of certain prosodic events might be tied to the structure of the syllable, as different parts of the syllable carry different amounts of prosodic information. Consider for instance the phrase “steak?”, where most of the prosodic information (i.e., the upward inflection) is carried by the vowel (i.e., the nucleus) and not the fricative (i.e., the onset).

This issue cannot simply be addressed by including positional information such as percentage values. Percentage values are not very reliable for syllables, because the internal structure of the syllable can change dramatically depending on the identity and relative duration of the phones it contains.

In order to better account for these timing phenomena and to ensure better alignment between the syllabic structure and the F_0 contour, I decided to include onset and rhyme labels.

2.2.3 Syllable

Because of the importance of syllabic units, I included labels for both lexical stress (i.e., primary, secondary, and no stress) as well as syllable boundaries (i.e., either present or absent).

2.2.4 POS tags

The next linguistic label I decided to include is POS tags. These are necessary in order to capture prosodic events pertaining to the syntactic domain. Phonologically identical words (e.g., the word “cut” can function both as a noun or a verb) often have different syntactic functions depending their POS tag and correspondingly different prosodic behaviors.

Intuitively, a verb like “to wrap” behaves more closely to other verbs than the word “wrap” used as a noun. Notice how in the following sentences we can switch the verbs coming after “want to” and maintain the same overall prosodic profile (provided the new word has the same accent groups).

- (1) Tonight I want to **make** a few presents.
- (2) Tonight I want to **wrap** a few presents.
- (3) Tonight I want to **send** a few presents.

2.2.5 Lemmata

The next set of labels I included in my implementation is word lemmata and word boundaries. This might seem like an unnecessary complication since word vectors will greatly increase the size of the model.

The reason why word information is important is that words (especially used in context) can carry with them the semantics of a sentence as well the underlying attitude of a speaker who might decide to speak such an utterance. For instance a speaker uttering “I am so depressed today!” is

more likely to use a generally flatter and monotonous tone than someone who’s uttering “I feel so alive today!”.

Another reason it is desirable to include word vectors is that a lot of syntactic classes described by POS tags are not fine-grained enough to capture important semantic and syntactic differences. For instance, adverbial expressions such as “additionally”, “clearly”, “not”, “then”, “yesterday” are all labeled as part of the same group, even though they display very different syntactic behaviors, which can often lead to very different intonation patterns.

In most language models, word vectors are not usually learned using vanilla softmax classification. This is due to the extremely large number of possible outputs (i.e., context words), which makes training very resource-intensive and time-consuming. A common workaround to this issue is to use noise classifiers (Mikolov et al., 2013). In our particular case, we will show that we have a fairly small number of output classes (for more details on the output of the proposed intonation model, see Section 2.3), which means we are able to train word vectors using standard loss functions.

As adding words embeddings to the model is computationally expensive, it is important to select and process the word tokens carefully. Morphological information is redundant, as it is by and large already encoded by the POS tags. Therefore it can be discarded by means of lemmatization.

Secondly, it is important to remove word lemmata that are only frequent as a mere consequence of the specific word distribution of the training corpus, e.g., character names, place names, etc. These words are only specific to the training corpus. It is therefore unlikely that they will come up again and that they will generalize well within new contexts. To remove these words, we filter out the lemmata that are not found in frequency lists of English based on much larger corpora.¹

2.2.6 Punctuation

The last feature I added to my implementation is punctuation. Punctuation, aside from helping distinguish between sentence types (interrogative, declarative, exclamatory, etc.) is a feature that is also useful for demarcating syntactic phrases and solving syntactic ambiguities. This in turn can help us decide which prosodic strategy to adopt.

Since punctuation does not have a reliable acoustic representation (occasionally realized through pauses), punctuation annotations are added in

¹The frequency list used to filter out words is derived from the the British National Corpus (<http://www.natcorp.ox.ac.uk/using/index.xml?ID=req>).

correspondence to the words. In particular, for every word, I added a label for punctuation before and after the word. In the case of punctuation before the current word, the punctuation label is generated by concatenating all the text characters between the current word and the previous word. Similarly, for the punctuation after the current word, the punctuation label is generated by concatenating all the text characters between the current word and the next word. White space interspersing punctuation characters is removed. The resulting punctuation labels are then augmented with a special punctuation label for no punctuation.

2.2.7 Input vectors

Linguistic labels cannot be used directly and fed to a neural network, as neural networks can only accept data in the form of input vectors.

In our case, most of our data is not sequences of numerical values, but rather categorical data such as words, stress information, POS tags, etc. In order to use this information, we first need to encode these values into one-hot vectors. In one-hot vectors, all positions contain zeros, except for the position referring to the observed label, which contains a one. This process converts the categorical data into binary features, that can be used by the neural network. Based on the corpus used in the implementation, the binary features reported in Table 2.1 were obtained (for more details see Appendix A and Appendix B).

Level	Description	Size
Word	lemmata	1937
	POS tags	66
	word boundaries	3
	punctuation before word	55
	punctuation after word	55
Syllable	syllable boundaries	3
	syllable stress	5
	onset/rhyme	3

Table 2.1: Linguistic features used as input for the F_0 -DNN model.

Binary features encoding various linguistic information can then be strung together into a single larger vector. This is what is usually done in most neural network applications.

In the proposed approach, I followed a similar approach, with the difference that I decided to separate word vectors from the other linguistic inputs (see Figure 2.5). This choice was primarily made for the sake of performance at inference time (for details, see Chapter 3). Word vectors have an extremely high number of binary features and constitute the most expensive part of the model.

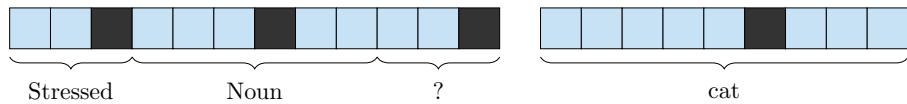


Figure 2.5: Example of an input vector. Word and non-word features are encoded by two separate vectors. This particular set of vectors could refer to a data point sampled at the end of a sentence such as “Where is the cat?”.

2.3 Intonation Labels

In addition to selecting the appropriate linguistic labels as input, one must also generate intonation labels for the output, so as to provide supervision during the training of the model. All the previous work on intonation modeling discussed in Section 1.2 focuses on a static description of the F_0 , either as a value (e.g., $\log(F_0)$, wavelets, quantized F_0 , etc.) or as a pattern (e.g., templates, splines, B-splines etc.). Here, I propose a dynamic approach, whereby contours are not encoded as a sequence of static values, but rather as a sequence of values representing the dynamic evolution of the contour through time.

In order to achieve this, a new encoding scheme for the interpolated F_0 has been devised. The encoding scheme is characterized by three key features: quantization, dynamism, and compression. Quantization is achieved by discretizing the frequency space into frequency levels. Pitch movements from one level to the next are encoded dynamically as sequences of pitch intervals, rather than sequences of static landing sites. Pitch intervals are compressed by representing intervals as the combination of two components: *sign* and *magnitude*. The sign represents the direction of pitch change, and the magnitude, the amount of change in such direction. The magnitude representation is further compressed by rounding magnitude values to their closest triangular number approximation.

2.3.1 Encoding

The prosodic dynamics are calculated from static values sampled from interpolated F_0 contours² by means of the following recursive formula:

$$f_t = \begin{cases} 2^{(n_t/s)} & t = 0 \\ f_{t-1} * 2^{(n_t/s)} & t > 0 \end{cases} \quad (2.2)$$

where:

t = time index

s = number of steps within each octave

n_t = number of steps away from f_{t-1}

f_t = frequency of the F_0 value n_t steps away from f_{t-1}

The formula automatically maps F_0 values to an underlying pitch scale, where each octave contains s equally sized steps (pitch levels). In my

²In my implementation, contours are estimated and interpolated by means of the software package Praat (<http://praat.org/>).

implementation, s is set to 24 in order to have each frequency interval correspond to exactly 1/2 semitone of the standard western scale. At each time step t , we estimate f_t to produce the closest approximation to the observed F_0 value at time t . The n_t value that produces the closest F_0 approximation constitutes the dynamic information we want to store.

As no values come before the first one, the estimation of $f_{t=0}$ is only needed to start the recursion for the subsequent f_t values. Therefore $n_{t=0}$ does not represent a dynamic, but rather the position of $f_{t=0}$ within the underlying scale (i.e., how many steps away from the bottom of the pitch scale). As neither an upward nor a downward pitch movement takes place before the very first F_0 value, $n_{t=0}$ is set to 0 after the recursion is complete.

The dynamic information n_t is separated into two components: the *sign* of n_t which gives the direction of change, and its *magnitude*, which corresponds to the amount of change in such direction. The sign is encoded by using three values: -1 (falling), 0 (level), and 1 (rising). Magnitude values are rounded to their closest triangular number approximation, where triangular numbers are defined as follows:

$$T_n = \sum_{k=1}^n k = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} = \binom{n+1}{2} \quad (2.3)$$

where:

$$\begin{aligned} T_n &= n^{\text{th}} \text{ triangular number} \\ \frac{n(n+1)}{2} &= \text{a binomial coefficient} \end{aligned}$$

2.3.2 Decoding

At decoding time, we face the inverse problem. We have a sequence of signs and magnitudes. From these, we want to reconstruct a sequence of F_0 values.

To recover the pitch interval information encoded by n_t , sign and magnitude are multiplied. The dynamics are recursively applied (i.e., added) to a starting seed value. To initialize the recurrence, the seed is set to 0.

This process automatically produces static values mapping to the underlying pitch scale's levels. To convert these static pitch levels into their corresponding Hertz values, a pitch scale is generated with the following equation:

$$f_n = 2^{(n/s)} \quad (2.4)$$

where:

s = number of levels within each octave

n = scale level

$f_n = F_0$ value at the n^{th} level of the scale

As the frequency mapping scale does not contain any negative values, we shift the static values to be decoded until they are all positive. Then, we produce an initial F_0 sequence by replacing each static value by its corresponding frequency generated by Equation 2.4. This sequence is then shifted to correspond to the speaker characteristics, using the following translation:

$$f_t^* = f_t - \frac{1}{k} \sum_{t=0}^k f_t + \overline{f(s)} \quad (2.5)$$

$\overline{f(s)}$ = average F_0 of speaker s

t = time index

f_t = F_0 value at time t

f_t^* = F_0 value at time t adapted to speaker s

2.3.3 Quantization

One major characteristic of the encoding scheme I have presented is its quantizing effect on the encoded contour, which transforms continuous curves into sequences of discrete categorical data.

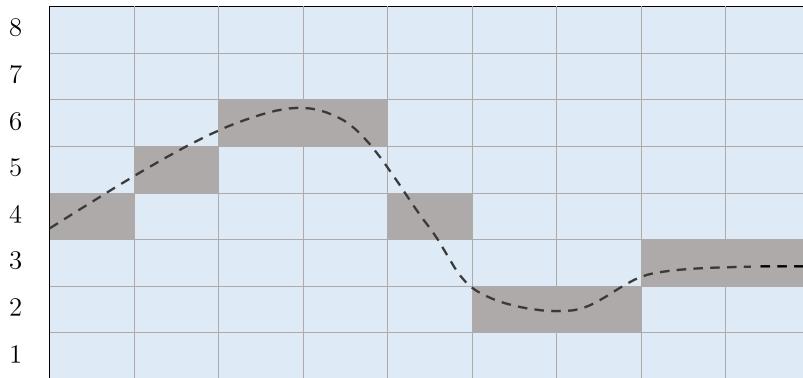


Figure 2.6: Example of a quantized contour. The F_0 contour is represented by a dashed line. The darker rectangles represent the corresponding quantized contour.

From the perspective of learning, quantization has the effect of converting F_0 modeling from a regression problem to a classification one. More specifically, under a quantized regime, a DNN would not attempt to approximate an underlying function of the contours, but rather a function of the probabilities of the categorical symbols from which the observed contour is generated.

This decision might seem rather odd at first, as most researchers usually choose to preserve the original continuous F_0 values and try to model the underlying function of the contour. I decided against this approach because in my estimation it might restrict and flatten the possible contours that the model can output. This is due to the fact that, as the model tries to reconstruct the underlying function, it will interpolate and smooth it to fit the training data. This has the effect of producing an averaged-out version of the observation.

In order to avoid this, I decided not to use numerical values and to replace them with categorical data describing quantized pitch jumps. As the categorical data represents clear pre-defined pitch movements and not static values, pitch values are not forced to fit into a smoothed function of the contour, but rather a function of the probabilities of the categorical symbols from which the static F_0 contours are generated. This means that the model is less restricted with regard to how fast pitch values can change or how far up or down they can jump, because the weights of the network are not modeling the actual size of the output values, but rather probabilities of pre-defined pitch movements.

The inclusion of quantization as a feature is also justified by the fact that F_0 is limited to a specific frequency range (typically 85 to 180 Hz for males and 165 to 255 Hz for females) and that very small pitch variations (say, 100 and 100.02 Hz) are not that important in the context of TTS, where reconstructed F_0 contours often only approximate the original. This is for instance the main assumption underlying the IPO intonation model (see Section 1.2 for more details), where contours are simplified to only preserve perceptually relevant prosodic features and where less important features such as micro-prosody are discarded.

In my encoding scheme, quantization is achieved by the recursive application of Equation 2.2, which automatically maps the static F_0 values to a quantized frequency space (i.e., a pitch scale). This equation is more commonly used to determine the frequency of notes of the equal-tempered western scale, i.e., a scale where each octave is divided into (twelve) equally-sized steps. In my implementation, the use of this formula was slightly different from its more common application, as it not used to directly partition the frequency space into levels, but rather to estimate the number of pitch levels separating one observed F_0 and next.

The choice of basing the encoding scheme on this particular scale is somewhat unconventional, as a more common choice in speech applications would be to utilize a mel scale. One reason behind this decision is the criticism surrounding the mel scale put forward by Donald D. Greenwood, a student of Stevens who worked on the mel scale experiments in 1956, who considers the scale to be biased by experimental flaws (Greenwood, 1997). Equation 2.2, on the other hand, is purely based on well-understood physics of sound, as it was not determined empirically through the judgment of test subjects.

Secondly, Equation 2.2 is adopted for the sake of convenience, as it much easier to debug and check the sanity of the encoding when the encoded values have a clearly interpretable meaning. For instance, if we set s to 24, each interval in the scale corresponds exactly to half a semitone of the standard western scale.

The idea of quantizing contours is not new in the context of F_0 modeling. In a recent proposal, quantization of F_0 into static discrete values has been adopted as a way of avoiding F_0 interpolation (Wang et al., 2017a). In particular, unspecified F_0 values (e.g., in the presence of unvoiced segments) are encoded by a dedicated categorical symbol. This is argued to help avoid the influence of prosodic artifacts created during the interpolation process. From the perspective of dynamism, this approach is functionally very similar to calculating the log of F_0 values: quantization

simply linearizes the logarithmic relationship between F_0 and pitch, with the difference that the static representations are discrete.

However, in the methodology proposed here, quantization carries out a completely different function, as it is not used to avoid interpolation, but rather to provide a quantized space as the basis for the calculation of the dynamics. In the proposed approach, it is not possible to do away with interpolation, because each F_0 label is defined in relation to the previous one, which means we can never leave the value of F_0 unspecified, otherwise all the following F_0 labels would become uninterpretable. Therefore, here, quantization of the frequency domain is just an indirect way of ensuring that dynamics are also quantized and not real-valued.

2.3.4 Static vs. Dynamic

The main feature of the proposed encoding is its dynamism. This means that contours are not encoded as a sequence of static values, but rather as a sequence of values representing the dynamic evolution of the contour through time.

This idea stems from a number of empirical and anecdotal observations that can be made about pitch both in the context of speech and music. In particular, the first observation is that raw F_0 measurements assign an absolute value to each sample, and this value is completely meaningless when taken in isolation. It is only when F_0 values are presented in context that they are endowed with prosodic meaning.

This closely mirrors a similar observation that can be made about the musical domain. In music, notes, just like speech pitch points, are completely meaningless by themselves. As most musicians will know, what in fact constitutes a melody is not so much the sequence of notes, but the sequence of musical intervals, i.e., the relative change between one note and the next one. This is especially apparent if we consider that most people who can sing a tune can do so without any knowledge about the name of the notes in the melody. Melodies are also routinely transposed (i.e., pitch translation) to suit various voice ranges without changing the musical message of the tune. All this can be explained by the fact that, to a large extent, humans, with the possible exception of those born with perfect pitch, do not rely on absolute measurements to produce and process musical pitch.

Similarly to what happens in music, humans do not know the exact F_0 values they process or produce (again with the exception of those with perfect pitch) and yet, they are still able to correctly generate and interpret intonation patterns. Furthermore, speech patterns can also be transposed,

i.e., when for instance they are uttered by people with different voice registers, without much change in the prosodic message (e.g., a question is interpreted as a question regardless of the shift of the contour along the frequency axis).

We do not wish to take this analogy any further. Speech and music are after all two completely different domains and there is even some evidence that the processing of pitch information differs significantly for speech and music phenomena. Zatorre and Baum (2012) suggest that there are two pitch-related processing systems in the human brain: one for coarse and approximate pitch analysis, and one for precise and fine-grained analysis. Of these, it is suggested that the latter is unique to music. Because of the significant differences between the two domains, within the scope of the present discussion, the only loose analogy that we wish to take advantage of is the fact that the encoding of both musical or prosodic messages relies on a somewhat more relative dimension of frequency than an absolute one.

The importance of a dynamic representation of the contour rests in its ability to capture a more abstract and compact representation, endowed with a higher degree of invariance. This property is illustrated by Figure 2.7. As it can be observed, the static values after the upward translation are completely different. On the other hand, a dynamic representation is completely invariant to frequency translation:

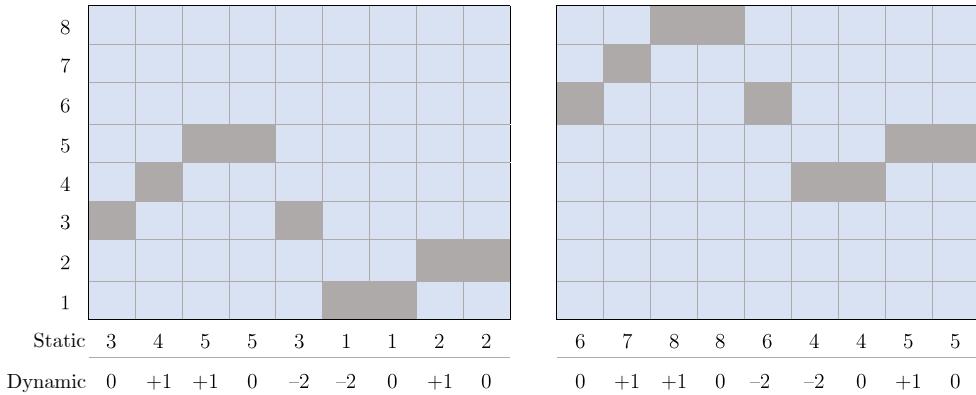


Figure 2.7: On the left side of the figure, a quantized F_0 contour is shown. On the right side, the same quantized F_0 contour has been shifted upwards by three steps. At the bottom of the figure, the corresponding static and dynamic values are reported.

As a side effect, the proposed encoding scheme is also naturally register-

independent, as any upward or downward translation of the original contour would yield the same encoded sequence. This means it can be used to seamlessly model intonation of multi-speaker corpora without any additional normalization steps.

2.3.5 Sign Separation

One important feature of the proposed encoding scheme is the decomposition of pitch intervals into sign and magnitude information. The reason for the separation of these two types of information is justified first and foremost as a way of reducing the number of possible outcomes by half.

One additional advantage of this separation is that we are also able to produce a representation that has a higher degree of invariance to compression and dilation, especially with regard to sign information. This property is illustrated by Figure 2.8, where we can see how the sign of a contour remains unaltered, even after dilation has taken place. We can also observe that the magnitude, similarly to traditional static values, is more susceptible to noise.

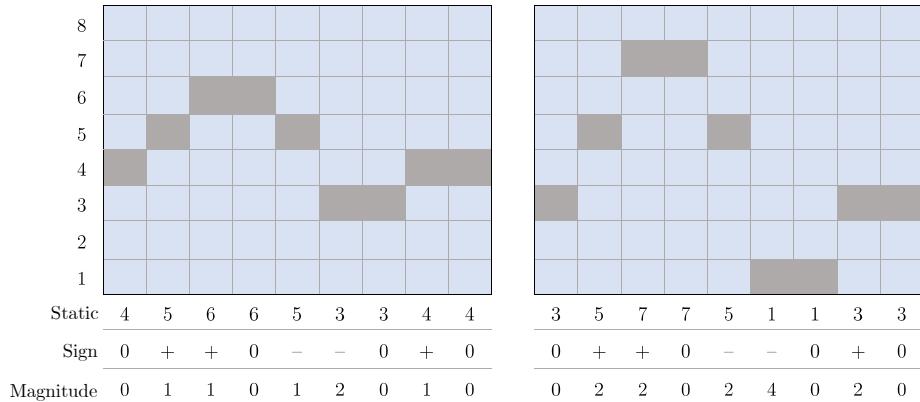


Figure 2.8: On the left side of the figure, a quantized F_0 contour is shown. On the right side, the same quantized F_0 contour has undergone dilation. At the bottom of the figure, the corresponding static and dynamic values are reported. The dynamic values are represented by the corresponding sign and magnitude values.

2.3.6 Magnitude Compression

The last main feature that characterizes the proposed dynamic approach is its compressing effect. Compression is achieved in a first instance by means of quantization, as this greatly restricts the possible number of outcomes. This is because the continuous infinite space is mapped to simplified quantized space with a discrete and finite number of possible output values.

A second source of compression is provided by the separation of sign and magnitude information. This atomizes the contour representation into its minimal components: the direction of movement and the amount thereof.

Even though separating the sign and the magnitude reduces the number of possible outcomes by half, the number of magnitude labels is still much too great and wasteful if not properly compressed. This is especially true as far as large intervals are concerned, where we might not want to model the small difference between extremely large interval values.

One way to reduce the number of labels, would be to use larger steps by setting the s parameter in Equation 2.2 to a lower value. This, however, would also reduce our ability to encode smaller prosodic movements, which would result in very imprecise contour reconstruction. Ideally, our encoding scheme should be able to model most prosodic events, so both small and large intervals.

However, we might not wish to model them with the same level of precision, since small differences between large intervals might be less important to model. If the processing of pitch information for speech is based on a fairly coarse-grained analysis of pitch as suggested in (Zatorre and Baum, 2012), encoding fine-grained differences is unnecessary.

For these reasons, a third source of compression is added. In the proposed encoding, the number of possible magnitude values is drastically reduced by rounding magnitude values to their closest triangular number approximation. Triangular numbers are a sequence of numbers that are obtained by continued summation of natural numbers.

For instance, the sequence of natural numbers “1, 2, 3, 4, 5, etc.” would yield the sequence of triangular numbers “1, 3, 6, 10, 15, etc.”. For my encoding scheme, an additional 0th triangular number is included, i.e., “0, 1, 3, 6, 10, 15, etc.”.

Triangular numbers have the nice property that the distance between each number and the next one grows linearly. When we only consider intervals whose index is a triangular number we are able to consider intervals with a linearly decreasing precision. This means that we are able to model small intervals very precisely, with the trade-off that we model

large intervals less precisely.

2.3.7 Encoded Contours

In order to assess the validity and usefulness of my encoding, I compared the original contours with their encoded-decoded counterparts by plotting and listening to their copy-synthesis.

Figure 2.9 shows an original contour along with its encoded and decoded counterpart. As we can see, the original and the reconstructed contours are quite similar, albeit with a few subtle differences. In particular, observe in Figure 2.9 how all the numbers of steps in each pitch interval are triangular numbers (i.e., 0, 1, 3, 6, 10, 15, etc.). For instance, the second syllable (i.e., “mid”) has two intervals, where the first one has 10 steps and the second one, 6. Also, notice how the reconstructed contour occasionally overshoots or undershoots. For example, in the fifth syllable (i.e., “naits”), the first pitch interval contains only 3 steps instead of the more accurate 4.

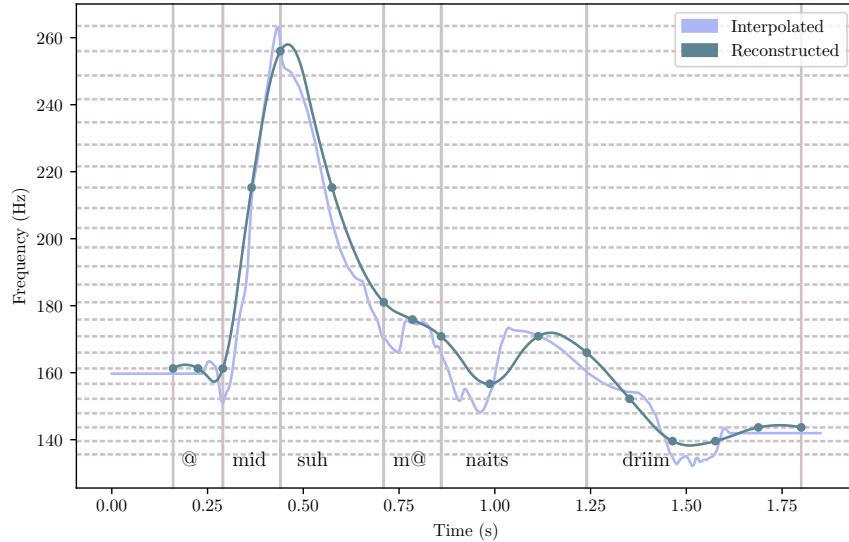


Figure 2.9: Plot of the original and reconstructed contours for the utterance “A Midsummer Night’s Dream”. The dashed horizontal lines represent the levels of the underlying pitch scale. The vertical lines represent syllable boundaries. At the bottom of each syllable boundary are the phones contained in the syllable. The dots represent the points in time where the F_0 values were sampled.

This is because this encoding scheme was not designed with reconstruction accuracy as its main goal in mind, as it was actually conceived of from the start primarily as a way of producing extremely minimal, flexible and dynamic representations of contours.

Even though we might be prepared to accept less than perfect reconstruction as a trade-off in favor of increased flexibility, it is still important to quantify the exact extent to which the original and reconstructed contours differ. A measure of distance between the original and the reconstructed contour can give us an approximate and indirect way of quantifying how much of the original prosodic message has been lost.

Despite the fairly noticeable differences observed in the plots of the original and reconstructed contours, the measured root mean squared error (RMSE) distance for the entire corpus came out to a fairly low 1.03 Hz. Pulkki and Karjalainen (2015) cite 1 Hz as the just noticeable difference (JND) threshold for a 250-500 Hz pitch range. This means that on average, the original and the encoded-decoded signals will largely be indistinguishable pitch-wise. This result was also confirmed, albeit only anecdotally, by listening to the copy-synthesis.

It should be pointed out that lossless or near-lossless encoding is not a necessary requirement, nor is it the primary aim of my encoding scheme. As pointed out previously, the feature we are interested in preserving is not exact F_0 measurements, but relative pitch changes with a strong emphasis on minimal and flexible encoding. However, given that the proposed encoding scheme can reconstruct the original F_0 measurements to a fairly high degree, we can rest assured that converting the original measurements into pitch intervals did not result in unexpected artifacts or excessive skewing of the data.

In order to better understand the effects of the encoding on the original data, I also looked at the distribution of the original F_0 values (see Figure 2.10) and the distribution of the reconstructed F_0 values (see Figure 2.11) for the entire corpus, after they were decoded from pitch interval sequences.

As we can see from Figure 2.10 and Figure 2.11, F_0 values from reconstructed contours retain the overall distribution of the original ones. The most obvious difference is that the reconstructed F_0 values cluster around the values found in the semitone scale underlying the quantization process.

One final aspect that I investigated was the pitch labels that were produced by the encoding process itself, i.e., before decoding them back into F_0 values, as these are the actual values that will be fed the neural network. After collecting all instances of sign and magnitude for the entire

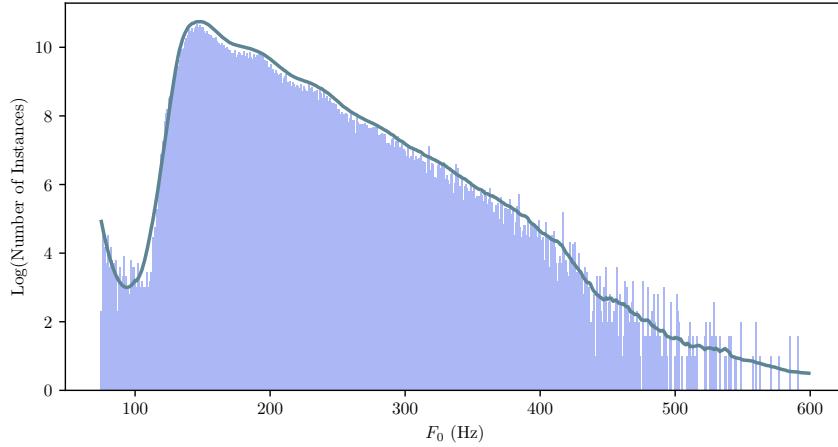


Figure 2.10: Plot of the interpolated F_0 values in the training set.

corpus, the two complete sets turned out to be the following:

- Sign: $-1, 0, 1$
- Magnitude: $0, 1, 3, 6, 10, 15, 21, 28, 36, 45, 55$

One striking aspect of this result is that the size of these two sets is extremely small, as only three labels for the sign and only eleven labels for the magnitude were used to reconstruct the original contours for the whole corpus. This is somewhat unexpected, especially considering the pitch variety that can be observed in Figure 2.11 and the relatively low RMSE distance between the original and the reconstructed contours.

This results highlights another feature of this encoding scheme that I had not originally considered, i.e., its highly generative nature. Just as a small set of morphemes can generate a virtually infinite number of words, it should also be possible to combine pitch movements to generate a virtually infinite variety of contours, among which natural ones hopefully also reside.

As we can observe in Figure 2.12, the pitch interval distribution seems to follows a power-law distribution, as it is expected in many frequency rankings in the linguistic domain. Also, notice how the distribution is mirrored and centered around the zero value; and how the intervals at either side of the plot are progressively further apart. This effect is due to the use of triangular numbers, which allow us to capture smaller intervals more precisely than larger intervals.

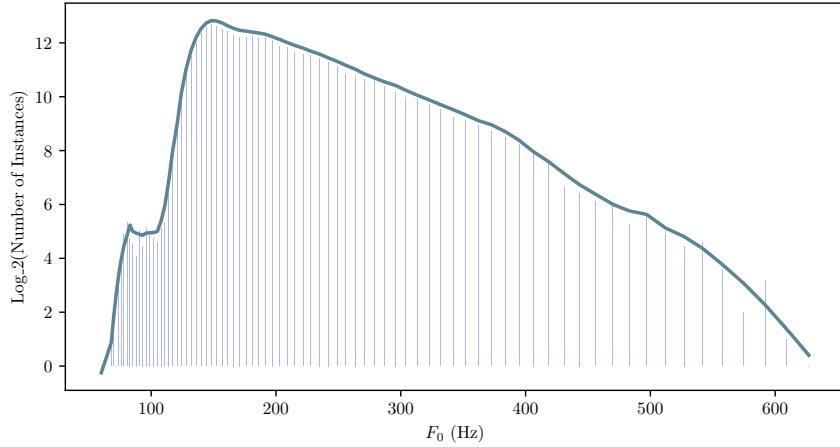


Figure 2.11: Plot of the F_0 values after they were encoded to pitch intervals and then decoded back into F_0 values.

2.3.8 Output vectors

In order to use the proposed encoding in the context of DNN models, it first needs to be converted into output vectors.

Sign and magnitude information is first converted into their corresponding binary features. Based on the corpus used for training, the output binary features shown in Table 2.2 are obtained (for more details see Appendix A).

Description	Size
Sign	3
Magnitude	11

Table 2.2: Intonation features used as output for the F_0 -DNN model.

These binary features are then converted into two separate binary vectors (shown in Figure 2.13) so that they can be fed to the DNN model. This way, one part of the network can attend to the modeling of the sign, while the other, the modeling of the magnitude.

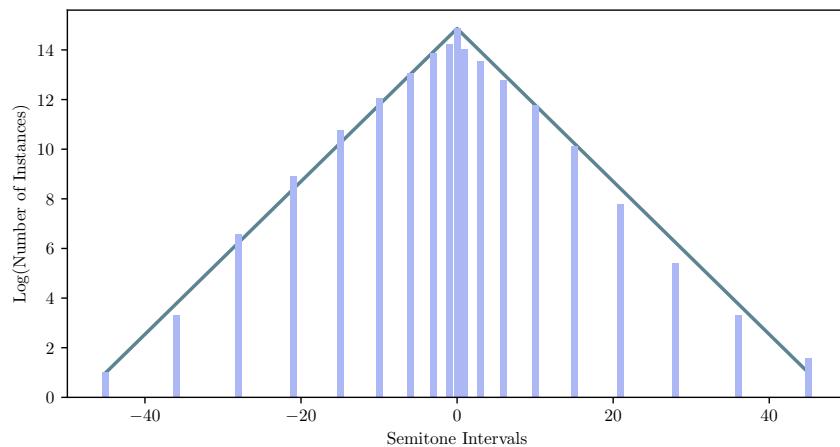


Figure 2.12: Plot of the distribution of the pitch intervals generated by the encoding process for the training set.

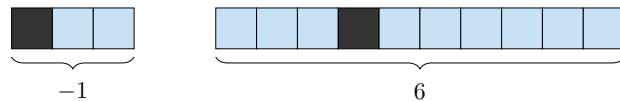


Figure 2.13: Example of an output vector. This particular set of vectors would refer to an output representing a six step downward pitch moment, where the “-1” label represents the direction of the movement and where the “6” label represents the number of step in that direction.

Chapter 3

Intonation Modeling

In this section, I shall provide a detailed description of the dedicated DNN model that I implemented for the prediction of intonation. The description of the inputs and outputs have been amply explained and justified in the previous sections, where I also showed how they are converted into suitable vectors that can be fed to the model. The input is comprised of two sets of one-hot vectors: one for the lemma information and the other for all the other textual information. The output also consists of two sets of one-hot vectors: one for the sign and one for the magnitude information.

Here, I will present the DNN architecture that is tasked with the mapping of the inputs to the outputs and I will provide the reasoning behind various implementational choices that were made, many of which were arrived at by trial and error after much experimentation.

In the first part of this section, I will provide a general overview of the architecture, including its main features and components. Subsequently, I will delve into various implementational and architectural choices and reasons in much greater detail.

3.1 General Overview

The DNN model, shown in Figure 3.1, is structured into three main components: a word embedding component, a linguistic interaction component and a prediction component.

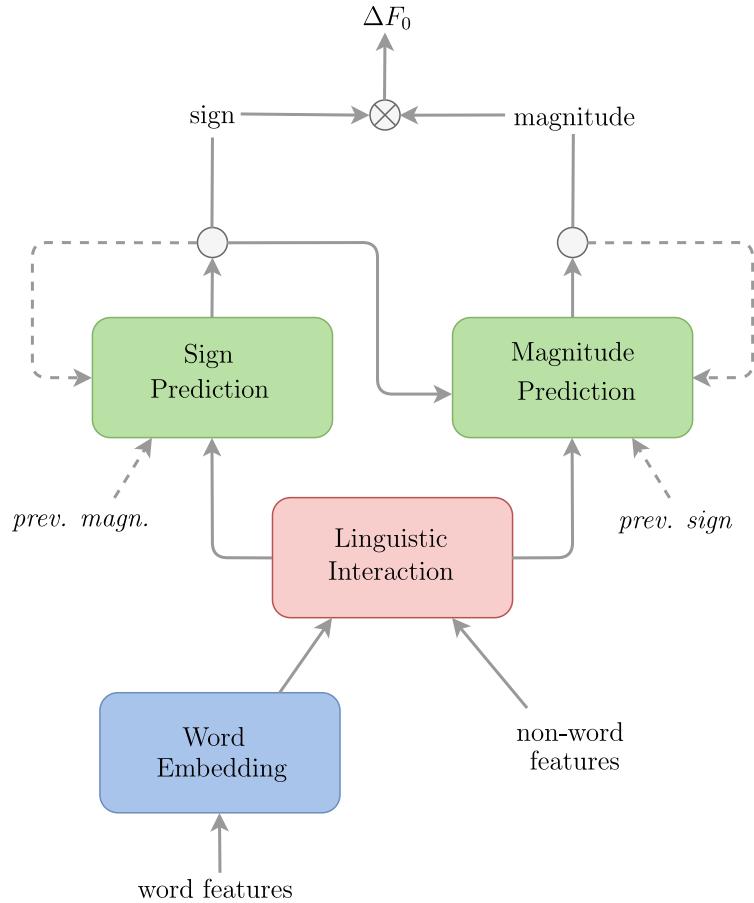


Figure 3.1: DNN pipeline. The dependencies on the previous states are represented by dashed arrows, and the forward dependencies, by plain arrows. The blue box corresponds to a FFNN, the red box, to a network composed by a feed-forward layer and a bidirectional recurrent layer, and the green box, to a network composed of a feed-forward layer and a forward recurrent layer.

The word embedding (Levy and Goldberg, 2014) component is comprised of three feed-forward layers of the following sizes: 2048, 1024, 512.

The main function of this component is to convert the one-hot word vectors into smaller, denser vectors by projecting the lemma information into a dedicated reduced space.

The next component is the linguistic interaction component. At this stage, the output of the word embedding component is concatenated with all the other linguistic inputs and passed through one feed-forward layer of size 512.

The purpose of this feed-forward layer is to merge the dense lemma representations and the other linguistic inputs. This gives inputs a chance to have some local interaction before they are fed to additional layers.

Once the inputs have been processed locally, i.e., within the time step they were emitted, they are fed to pair of BRNN layers with 512 GRU neurons each. The purpose of the bidirectional layers is to capture contextual interaction through time.

The last component is composed of two similar sub-components. These are tasked with the prediction of the sign and magnitude of the dynamic of the F_0 , respectively. Each sub-component is composed of a simple feed-forward layer of 512 nodes and a forward recurrent layer of 256 GRU neurons, to take the context into account.

The input to this last component is the backward and forward states produced by the previous component, plus its own recurrent sign and magnitude states. In order to align the sign and the magnitude predictions, the magnitude sub-component has additional input coming from the output of the sign sub-component. ELU (Clevert et al., 2015) is used as the activation function for all neurons in the model.

3.2 Architectural Details

3.2.1 Word embedding

In the proposed approach, the training of the word vectors is integrated with the rest of the linguistic features within a single common structure. The reason behind this choice is that neural networks can model very subtle interactions and detect very fine-grained correlations. By training word vectors separately, the network cannot guide the dimensionality reduction on the basis of the interactions that these dense word vectors will have downstream with other inputs. By training all the labels within the same instance, all the interactions between linguistic labels and output labels are allowed to flow back into the feed-forward word layers.

However, because of the size of the word inputs, integrating the train-

ing of word embeddings within a common structure can cause significant performance degradation at inference time. To overcome this, word vectors are given a number of dedicated feed-forward layers, to project the large input vectors into smaller dense vectors. After the network has been trained, the dense word vectors can be extracted by feeding the one-hot word vectors one after the other. For each word we can then push the input thought network right until the last feed-forward word layer of the word embedding component. This last forward layer can be saved and used during inference in lieu of the original one-hot word vectors, thus bypassing the computationally expensive word embedding component.

3.2.2 Linguistic interaction

The linguistic interaction component is tasked with modeling both the local interaction between word and non-word features, but also contextual interactions across multiple time steps. In the proposed approach, these temporal interactions are modeled by means of BRNN layers. However, this is not the only possible approach, as these could also be modeled with feed-forward layers.

FFNNs have successfully been used in Vainio (2001) to model time series in the context of intonation modeling. In their work, the use of feed-forward layers is appropriate, as the author explains that their intention is not to model trajectories or curves directly, but rather instantaneous values within an utterance.

Under their approach, contours are not treated as events that unfold through time, but rather as something more akin to “static pictures”, where trajectories are just a side-effect of correctly predicting the static values. The task of the network would then be to match snippets of linguistic inputs to snippets of contour “pictures”. Given their initial assumptions and the way their work was set up, it makes sense to use feed-forward layers.¹

However, within the proposed methodology, this approach does not seem to be appropriate, as the proposed DNN is based on a completely different set of assumptions, most of which are antithetical to those underlying their work. In particular, the main assumption underlying my proposal is that contours are not to be viewed as sequences of “static pictures” that we try to fit together in a sequence, but rather as sequences of instructions of interpretable prosodic commands.

Even though FFNNs can in principle learn time series by using a sliding

¹or even convolutional layers, I would venture to suggest.

window, a more common and elegant way to approach tasks involving time series or forecasting is to use RNNs. RNNs are especially suitable for the modeling of time series, as they can be fed arbitrarily long sequences of data, whereas FFNNs can only model fixed size windows. Once the size of the window has been decided, we cannot feed inputs whose size is larger than that of the window. Similarly, inputs of smaller sizes either need to be concatenated with other inputs to fill up the input window, or they need to be padded. Given these limitations associated with FFNNs, in the proposed model, interactions over time are modeled by means of recurrent layers.

As RNNs come in different flavors, it is important to select the most suitable architecture for the task at hand. Vanilla RNNs are not optimal for my task, because they can only read the inputs one after the other, with the result that our context is limited to only past observations.

Even though humans almost never read the entire sentence before reading it out loud, they typically do look ahead and make use of many direct and indirect clues that the network has no access to. For instance, humans can extrapolate a lot of information about the general length and structure of a sentence or a paragraph by noticing punctuation symbols, line breaks, etc. For all these reasons, it would be desirable to present the neural network with both past and future inputs.

This can be achieved using BRNNs, where one recurrent layer reads the input forwards and the other backwards. Additionally, Schuster and Paliwal (1997) have shown that BRNNs achieve better performance than vanilla unidirectional RNNs.

3.2.3 Feed-back Connections

The last component of the the proposed DNN architecture is characterized by a rather complex structure, with multiple inputs and recurrent feed-back connections feeding into it.

This is in contrast with many other approaches, where the output of BRNNs is fed directly to the output layer to produce a prediction. This approach would be more than sufficient if we were in the situation in which the input features can only map to one possible output sequence. However, in many sequence-to-sequence problems this is often not the case. Consider for instance the case of machine translation (MT): for every sentence in the source language there are potentially multiple ways in which it can be rendered in the target language.

In this scenario, the task of the neural network is twofold: on the one hand, it has to predict an output sequence that would be fitting for

the input sequence; on the other, it has to make sure that each possible rendition of the input sequence is cohesive and coherent within itself. This means that in the event in which for some input sequence there are two possible output sequences that the network could predict and these two sequences are just as likely, we want the neural network to commit to one and only one of the two possibilities and not to blend different elements from the two that might not coherently fit together.

In MT, this problem is often solved by means of feedback and attention mechanisms (Cho et al., 2014; Luong et al., 2015). Attention mechanisms make sure the network learns to which parts of the inputs it should attend at every step of the prediction. This is especially important when the inputs are temporally not aligned with the outputs (e.g., when translating from English to German, the verb sometimes has to move to the end of the sentence).

In our particular case, an attention mechanism is not necessary, because our inputs and outputs are already aligned along the time domain. So, to make sure predictions were coherent, feedback connections from previous outputs are sent into current processing layers. This technique has also been used in the context of MT to ensure local fluency (Cho et al., 2014).

The introduction of feedback connections is also justified by evidence about the importance of feedback in speech (Borden, 1980). Particularly in what concerns intonation, it has been observed that, in postlingually deafened adults, intonation patterns undergo a progressive process of deterioration when feedback weakens as a consequence of hearing loss (Lane and Webste, 1991). Alterations in pitch pattern production are also observed in adults with unimpaired hearing in response to manipulated pitch feedback (Burnett et al., 1998). Adding feedback mechanisms to our network serves as a way to simulate the ability that humans have to hear and utilize their own previous pitch patterns in the generation of subsequent ones.

In order to model sign and magnitude simultaneously, both predictions are integrated within a single shared task. In ML, this technique is commonly referred to as multi-task learning (MTL). Aside from the ability to share tasks, there are many additional advantages in using MTL: it promotes selection of representations that are useful for all tasks, it allows for transfer learning, it can help focus attention on features that are most relevant, and it provides strong regularization (Ruder, 2017).

In order to add this sharing-task feature to my network, I first added two parallel feed-forward layers (i.e., the output of one feed-forward layer does not feed into the other), one for the sign and one for the magnitude,

where each has its own set of parameters. Each of the two feed-forward layers takes as input the concatenated current forward and backward states from the bidirectional layers, plus the sign and magnitude outputs predicted at the previous time step. Then, in order to model sign and magnitude interactions across time, I added one recurrent layer for the sign and one for the magnitude. Just like the feed-forward layers on top of which they sit, these two recurrent layers run alongside each other, and each has its own set of parameters. One of these recurrent layers makes predictions for the sign, and the other, for the magnitude.

At each time step the layers involved in the prediction of the sign receive information about the sign and magnitude that have been emitted in the previous time step. Likewise, the layers involved in the prediction of the magnitude also receive information about the sign and magnitude that have been predicted. This makes sure that the two sets of layers involved in each task can realign themselves to the other at each time step.

However, even with feedback connections that allow for realignment, we might still have problems of inconsistency within each time step. This is because the layers involved with the prediction of, say, the magnitude receive only information about what sign and magnitude have been emitted in the previous step, but they do not have access to what the current sign prediction is. To reduce the chance of inconsistencies within each time step, a connection between the current sign prediction and the current magnitude forward layer has been added. Under this approach, we assume that the task of sign prediction takes precedence over the task of magnitude prediction. The idea is that we first decide in which direction we want to move and then we can decide exactly the amount by which we move into that direction.

One important implementation detail is that, during the back-propagation stage, we must not propagate the magnitude error through its connection to the current sign prediction. If we neglect to take this crucial step, the error from the magnitude will flow into the layers dedicated to the prediction of the sign, which can cause the magnitude error signal to completely take over the sign representations so that they become predictive of the magnitude. Not only would this defeat the whole purpose of dividing the network into two tasks, but, in my experience, it can often lead to gradient-related errors.

In conclusion, at each time step, the prediction of the sign is conditioned on all the previous, current, and subsequent linguistic inputs, as well as the previous sign and magnitude predictions. The prediction of the magnitude, on the other hand, is conditioned on all the previous, current,

and subsequent linguistic inputs, as well as the previous sign and magnitude predictions, as well as the sign predicted at the current time step. The predictions that the network emits for both the sign and the magnitude will be used together with the output vectors that we feed to the network as to compute an error function describing the distance between our prediction and the training data.

3.2.4 The Error Function

There are two main error functions that are very popular in the context of deep learning: one is the RMSE and the other is the cross-entropy error.

As my problem is not formulated as a classification task, the cross-entropy error function is used. Even though it is theoretically possible to perform classification using the RMSE error function, it has been found that the cross-entropy function leads to faster training and better generalization in classification tasks (Golik et al., 2013; Simard et al., 2003).

3.2.5 The Optimizer

In contrast with recent and popular trends of using adaptive per-parameter learning methods such as Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), Rmsprop (Dauphin et al., 2015), Adam (Kingma and Ba, 2014) and Nadam (Dozat, 2016), I decided to adopt a variant of the classical stochastic gradient descent (SGD) algorithm.

Recently, it has been found that, for over-parametrized problems, as it is the case for most current deep learning applications, adaptive optimization methods can produce drastically different solutions, as well as models that generalize significantly worse than SGD, even when they perform better on the training data (Wilson et al., 2017). This observation was also at the basis of one of the recommendations offered by Hoffer et al. (2017) on how to train models that can generalize better.

For these reasons, in my implementation I used SGD with Nesterov Momentum, also called Nesterov accelerated gradient (NAG), which is a faster variant of Momentum optimization (Nesterov, 1983).

3.3 Accuracy Issues

One issue that is particularly hard to address in intonation modeling is how to measure accuracy during the training phase and what stopping criterion should be used.

When intonation is treated as a regression problem, i.e., when using $\log(F_0)$ as input and RMSE error as the error function, the obvious solution would be to simply use the loss on the function as a measure of accuracy. By calculating the loss on both the training and the validation set we can stop training as soon as the loss on the validation set starts increasing. This is a commonly employed technique also known as early-stopping.

The problem is that intonation is a highly unpredictable phenomenon. For every sentence, multiple completely different, yet plausible, renditions are possible. Which one of these should be predicted is highly dependent on a number of contingent factors (e.g., linguistic context, the speaker’s psychological state, the addressee, etc.), many of which are not available to the network. This means that at some point of the training process, we might produce contours that sound natural because they capture general prosodic properties from the corpus very well. However, the loss from these contours might be very high because the reference sentences just happen to be a completely different and equally plausible rendition of the input. A consequence of this possibility is that we cannot heavily rely on the loss of the neural network to measure the progress on the training.

As we currently do not have a reliable method to automatically measure the appropriateness of a specific contour against a specific set of linguistic inputs, the strategy adopted in my methodology was to save a copy of the model at the end of each epoch. After training, I selected one by listening to the synthetic contours it generated for the validation set.

3.4 Overfitting Issues

The reason why it is so important to determine the most appropriate time to stop the training of the network is because models produced at different epochs will be radically different.

If we train the model for too long, we start capturing too many details about the training data, many of which are specific to the training set and will not generalize well for new observations. This phenomenon is due to fitting the training data too closely and in machine learning it is also known as overfitting.

The chance of overfitting is particularly high in my model, because of the high number of parameters and the inclusion of very large layers for word vectors. One commonly used technique to alleviate this problem, is dropout. With dropout, the contributions of a subset of the neurons during training (Srivastava et al., 2014) are temporarily and randomly excluded. This process introduces noise and temporarily prevents some of

the inputs from reaching all the nodes in the network, thus reducing the chance that complex co-adaptations might emerge and forcing the network to make predictions only from partial sources of information. Despite its simplicity, dropout is very effective at reducing overfitting and is one of the most common regularizing techniques used in deep learning.

Because of these reasons, I added dropout to all the feed-forward layers of the network. This substantially alleviated the overfitting problem, which resulted in more consistent models across training epochs.

In previous sections, feed-forward layers preceding recurrent layers were justified as locations in the network to allow for local interactions between inputs. In addition to this main reason, feed-forward layers were also used to provide a convenient location to apply dropout.

In my architecture, there are many locations where one-hot vectors are concatenated with dense vectors before they are processed by recurrent layers. For instance, non-word linguistic inputs are concatenated with feed-forward layers before they are passed through bidirectional layers.

As we cannot apply dropout to very sparse vectors such as the one-hot vectors (because most of their inputs are already mostly zeros), feed-forward layers provide for a convenient location where we can merge sparse inputs with dense inputs to create a single dense representation, onto which dropout can be applied.

3.5 Data augmentation

Even though dropout produced a substantial improvement in the quality of the synthetic contours and greatly increased consistency across models, many of the predicted contours still contained extreme and implausible pitch excursions or pitch drops.

My hypothesis for this behavior was that the network was suffering from the so-called *exposure bias* problem. Exposure bias refers to a phenomenon often observed in generative models where outputs are fed back and used as inputs for subsequent output generation.

As explained by Ranzato et al. (2015), the problem is that during training, the model only utilizes the gold-standard output as its feedback and not its own predictions, which at inference time are bound to contain some errors or deviations from the original distribution. This discrepancy makes prediction brittle, as generation errors may accumulate over time.

One way to solve this issue is to randomly expose the network to either the ground-truth or its own predictions during training. However, this approach has been argued to be theoretically flawed (Huszár, 2015) and

not particularly successful in the context of F_0 modeling (Wang et al., 2017a), where instead one of the proposed solutions is to use dropout to occasionally remove feedback. However, in my network I already had dropout applied to the feedback connections and although dropout did help improve performance, it never completely eliminated the problem.

One thing I noticed was that often, the most problematic utterances were also the longest. On the one hand, this behavior is expected, as it is consistent with the exposure bias hypothesis: longer utterances allow for a larger margin of error, as we have more steps to predict and therefore more opportunities for generating errors. On the other hand, I could not understand why, once the prediction starts moving in the wrong direction, the network is unable to detect it and correct it.

It is easy to understand why generation errors are very problematic in language modeling, where interactions between words may have very unpredictable effects and errors are very hard to recover from. In humans, recovering strategies from these kind of errors is anything but trivial, as it often involves cognitively challenging tasks such as detecting the problem in the first place, rephrasing the problematic segment, or finding a convoluted and yet graceful way of still completing the utterance. These are all tasks that most neural networks are not trained to perform.

In our case, the property that we want to capture, and that is required to avoid getting out of a certain range, is a lot coarser and easier to define. This property is very apparent when we observe the F_0 contour over very long stretches of speech such as in Figure 3.2: pseudo-periodicity.

Even though over the course of a single utterance, a typical F_0 contour might display a declining (i.e., in the case of a declarative) or rising (i.e., in the case of questions) trajectory, over the course of arbitrarily long sequences, F_0 contours are pseudo-periodic, i.e., on average they move neither up nor down, as the F_0 values simply oscillate around a mean value.

This property is not immediately obvious if one only looks at the distribution of the static values shown in Figure 2.10. However, if we look at Figure 2.12, we can clearly see that the distribution of the pitch intervals is mirrored around the centre (zero). As we can see, for each positive interval there is a negative one, which means that over the course of a sufficiently long stretch of time the contours behave in a pseudo-periodic fashion.

My explanation as to why the network fails to capture this behavior is that the corpus is composed of mostly short utterances. The most common patterns in most training utterances are F_0 contours that either follow a globally declining (in declarative) or rising trajectory (in ques-

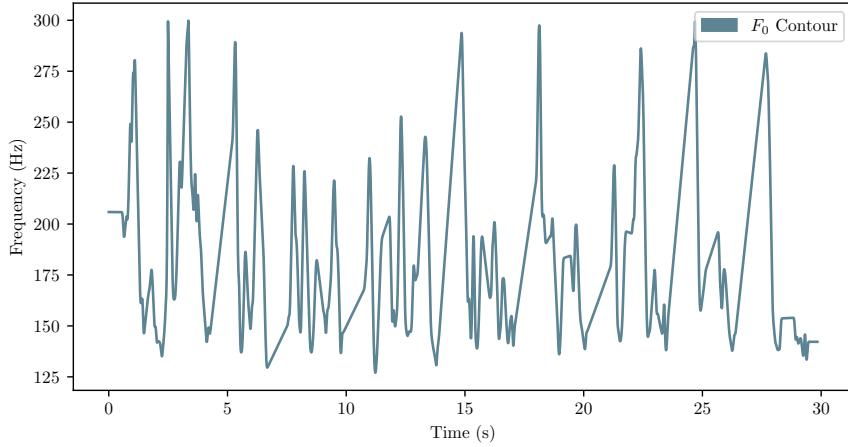


Figure 3.2: F_0 contour of a long (30 s) stretch of speech from the corpus. Notice how the signal presents a pseudo-periodic behavior: even though it is not periodic, peaks and valleys appear at fairly regular intervals.

tions), but only a few longer ones display the pseudo-periodic property I just described. So what might happen when the network is expected to predict unusually long sequences is that it will keep producing the the same declining or rising trajectory observed in shorter sentences, even though this entails moving outside the human voice range.

To fix this, the data was augmented by stitching contiguous utterances back together. The idea is that, by doing so, each training utterance will be much closer to the global distribution of the whole corpus. As a consequence, even if mistakes are produced during generation, the network will attempt to make use of the incorrect histories to still produce a pseudo-periodic behavior, thus greatly reducing the chance of moving outside of the human voice range.

This technique, in conjunction with dropout, proved to be a very effective way of alleviating both the problem of extreme pitch excursions/falls, as well as the out-of-range contours. After applying these techniques, the distribution of the static F_0 values and pitch intervals the test data (Figure 3.3 and Figure 3.4, respectively) was fairly similar to the corresponding distribution for the training data (Figure 2.10 and Figure 2.12, respectively).

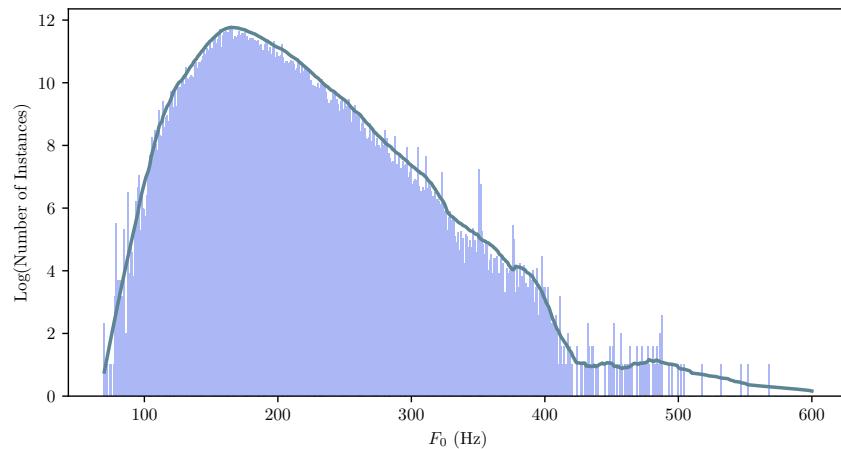


Figure 3.3: Plot of the interpolated F_0 values predicted for the test set.

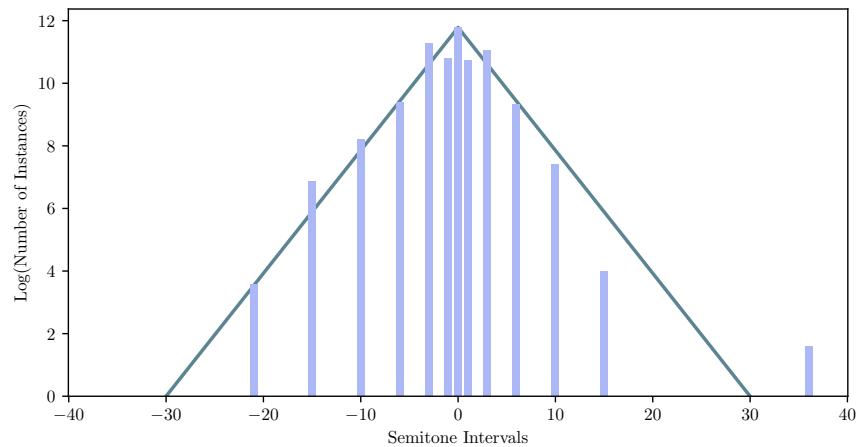


Figure 3.4: Plot of the distribution of the pitch intervals generated by the encoding process for the test set.

Chapter 4

The Segmental Synthesizer

In the first stages of my implementation, synthetic contours produced were simply transplanted onto the original signals by means of pitch manipulation in Praat. As this approach results in inconsistent quality across signals, a vocoder was used to synthesize the signals from scratch.

Even though vocoders can theoretically separate the periodic information (i.e., the F_0 contour) from spectral features and aperiodicity, in reality most vocoders (in particular those based on the source-filter model) do not achieve perfect decorrelation.

This means that each set of features cannot be modeled independently of each other without significant signal degradation. In order to model F_0 independently, the generation of aperiodic and spectral features must be conditioned on the periodic features.

This was achieved in my implementation by a neural network model that takes a linguistic specification along with duration and F_0 information as input, and predicts the remaining acoustic features as output. By conditioning the training of the acoustic features not only on duration and linguistic features, but crucially also on the F_0 , we are able to model the residual correlations left over by the vocoding process. The segmental synthesizer provides a way of inhibiting the effects of the spectrum, thus making the synthesis of spectral features consistent across many possible F_0 contours.

4.1 General Overview

The pipeline of the segmental synthesizer (shown here as Figure 4.1) is organized into two major components: a DNN component (the blue and the green boxes) and a waveform generation component (the red box). The DNN component is composed of two major components: a frame prediction component (the blue box) and smoothing component (the green box).

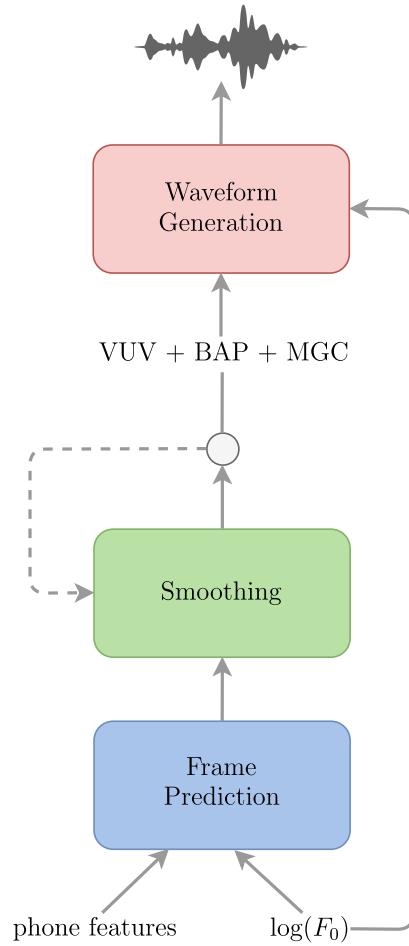


Figure 4.1: Pipeline of the segmental synthesizer. The Frame Prediction component (the blue box) is a 10-layer FFNN. The Smoothing component (the green box) is a single-layer RNN. The Waveform Generation component (the red box) is a vocoder.

The segmental synthesizer takes a linguistic representation (i.e., phone features) and $\log(F_0)$ as input. This information is fed to a FFNN to

predict a frame at each time step.

In order to ensure that the transitions between the frame are smooth, the output of the FFNN is passed through an RNN.

The output of this component, as well as the F_0 , are used by a vocoder to generate a waveform.

4.2 Description of the Input

According to common practice, input frames are produced at 5 ms intervals. The features contained in each frame consist of 225 binary features and 3 numerical features (for more details, see Appendix C).

The binary features, which constitute the linguistic specification, are used to encode quinphone identity, i.e., information about the current phone, the previous one, the next one, the one before the previous one, and the one after the next one. The phone set is based on the Oxford Advanced Learner's Dictionary (OALD) provided with Festival¹, augmented with an extra symbol for silence.

The numerical features comprise: a percentage value to encode positional information within the syllable, phone duration encoded as number of frames divided by 100 to make it fit approximately within a 0–1 range, and finally, $\log_2(F_0)$ divided by 10 to make it fit approximately within a 0–1 range. During training, the F_0 information is extracted by the WORLD vocoder and linearly interpolated.

Input vectors generated by the concatenation of the binary and numerical features are normalized to a standard normal distribution.

4.3 Description of the Output

For each input frame a corresponding output frame containing acoustic features is constructed. The output features contained in each frame consist of 2 binary features and 67 numerical features (for more details, see Appendix C).

The binary features encode voiced-unvoiced (VUV) information. The aperiodic and spectral information is extracted by the WORLD vocoder and then converted into 5 band aperiodicity parameters (BAP) and 60 mel-generalized cepstrum (MGC) coefficients, including the corresponding gain, using Speech Signal Processing Toolkit (SPTK).² The binary and

¹<http://www.cstr.ed.ac.uk/projects/festival/>

²<http://sp-tk.sourceforge.net/>

numerical information is concatenated into a single vector and then normalized to a standard normal distribution.

4.4 Description of the DNN Model

The neural network model is organized into two major components. The first component (shown as the blue box in Figure 4.1) is comprised of 10 feed-forward layers, each containing 1024 hidden units. These layers are tasked with the processing of frames within each time step.

For this first component, a special type of self-normalizing feed-forward layers are used. In self-normalizing neural networks (SNNs), the provided activation function is modified slightly so that neuron activations will automatically converge towards zero mean and unit variance. SNNs have been shown to outperform traditional FFNNs (Klambauer et al., 2017). In my implementation, the self-normalizing modification is applied to the ELU activation function (Clevert et al., 2015).

The second component (shown as the green box in Figure 4.1) is a single forward GRU recurrent layer with 1024 hidden units and ELU activations. The main purpose of this component is to replace the smoothing function of the static values across frames traditionally achieved through the use of the dynamics.

The network was trained using the sum-of-squares error function, which was minimized using SGD with Nesterov Momentum.

4.5 Wave Generation

During inference, input vectors are constructed with a synthetic F_0 contour.

The input vectors are passed through the network to generate VUV, BAP, and MGC coefficients. VUV information is used to set the unvoiced parts of the F_0 contour to zero. BAP and MGC coefficients are converted back to aperiodic and spectral information.

The F_0 contour, the aperiodic, and the spectral information are fed to the waveform component of the pipeline (shown as the red box in Figure 4.1) to produce an acoustic signal by means of the WORLD vocoder³.

³<https://github.com/mmorise/World>

Chapter 5

Evaluation

After the proposed methodology was implemented, a dedicated evaluation protocol was devised to evaluate the proposed model in comparison to a state-of-the-art parametric TTS system. In this chapter, I describe how the stimuli for the evaluation were prepared, the evaluation protocol, the results of the evaluation, and a discussion.

5.1 Stimuli Preparation

To run the evaluation, the 2017 Blizzard Challenge¹ speech corpus² was used. The full corpus comprises about 6.5 hours of speech taken from a number of children’s audiobooks, all read by a British female speaker.

The corpus was automatically aligned and pre-processed with the Montreal Forced Aligner (McAuliffe et al., 2017) and the MaryTTS system (Le Maguer and Steiner, 2017), producing 3866 utterances for a total of 3 h and 57 min of speech. The corpus was then split into three subsets: a training set (3475 utterances), a validation set (211 utterances), and a test set (180 utterances).

For the evaluation, three sources of intonation were considered: the intonation estimated from the original recording, the intonation produced with the proposed methodology, and finally, the intonation produced by the Merlin toolkit³ (Wu et al., 2016). The Merlin toolkit makes for a

¹For details on the Blizzard Challenge see <https://synsig.org/index.php/BlizzardChallenge2017>

²The corpus, originally made available by Usborne publishing (<https://usborne.com/>), is available at <http://www.cstr.ed.ac.uk/projects/blizzard/2017/usborneblizzard2017/>

³The implementation of the Merlin toolkit is available at: <https://github.com/CSTR-Edinburgh/merlin>

particularly apt comparison, because it also uses DNN technology, albeit in a very different way, as intonation and other acoustic features are modeled jointly.

As the main objective was to evaluate intonation, all stimuli used in the evaluation protocol were first neutralized with respect to duration and spectral features, to ensure that stimuli produced by different systems are comparable with respect to their intonation.

Duration was neutralized by imposing the duration of the original recordings extracted during the forced-alignment step. As for the neutralization of the spectral features, it was not possible to use the acoustic features extracted from the original recordings: even though vocoders based on the source-filter model can theoretically separate the periodic information from the spectral features and the aperiodic features, in reality vocoders often do not achieve perfect decorrelation of these sets of features. As consequence, replacing the estimated F_0 of a recording with an arbitrary one will most likely result in substantial quality degradation.

In order to inhibit the effects of the spectrum, a DNN segmental synthesizer was implemented. The segmental synthesizer used for the stimuli preparation is described in detail in Chapter 4. For the training of the segmental synthesizer, the same training, test, and validation sets were used. The mel-cepstral distortion (MCD) of the synthesizer measured on the test set was around 7 dB.

5.2 Evaluation protocol

As currently no objective evaluation can reliably assess the quality of the intonation of speech, a dedicated evaluation protocol based on subjective evaluation was devised. Subjective evaluations are primarily designed to probe more subjective and intangible aspects of speech that cannot easily be tested by means of objective evaluation, such as intelligibility and naturalness.

The dedicated protocol for the evaluation is a classic preference AB test, i.e., subjects are exposed to a pair of stimuli and they are required to express their preference. This choice is justified by the need to avoid the ceiling effect, as well as the lack of reproducibility and interpretability that often affects evaluations based on absolute judgments such as mean opinion score (MOS). The AB preference task was also chosen for its simplicity. As the test would be delivered online to people who might have never participated in speech evaluations, more common but also slightly more complicated evaluation protocols such as the multiple stimuli with hidden

reference and anchor (MUSHRA) test were deemed unsuitable.

A no-preference option was not included, lest subjects should abuse this option as a way to avoid expressing strong opinions. In order to prevent this from happening, during the evaluation, participants are always forced to make a decision. Later on, a lack of preference can be inferred by simply looking at preference distributions. For instance, if participants expressed preference for a system over another system within a certain percentage range, say 40%–60%, then we can assume there is no preference.

The evaluation was automatically prepared and managed by the online platform PercEval.⁴ The platform makes sure that stimuli are properly shuffled and spread out across participants in a statistically balanced way, so that all participants are exposed to as wide a variety of stimuli as possible and that, on average, all stimuli are seen a similar number of times.

From the test corpus, 180 distinct pairs were constructed. After providing the pairs and a set of configuration parameters, the platform started generating experiments on-demand. Whenever a new participant was available, a new experiment was generated and delivered online.

At the start of the evaluation, each participant was required to provide an email address and to take the brief questionnaire shown in Figure 5.1.

What is your age range? 15-20 ▾

What is your gender? Male ▾

Are you an english native speaker? Yes ▾

Are you a speech scientist? Yes ▾

How often do you listen to audiobooks?

At least once per day ▾

How often are you using a speech assistant (Siri, Google Assistant, Alexa, Cortana, etc.)?

At least once per day ▾

Are you using headphones for this evaluation? Yes ▾

Figure 5.1: Questionnaire displayed before the evaluation.

After the questionnaire, participants were introduced to a quick simulation, in which 6 of the total 180 pairs were used to allow them to familiarize themselves with the evaluation environment.

Of the 174 remaining pairs, each participant listened to 60 pairs, each selected randomly using a balanced random selection algorithm. At each

⁴The platform was made available courtesy of the EXPRESSION group at IRISA, <https://www-expression.irisa.fr/>.

step of the evaluation (here shown in Figure 5.2), subjects were presented with a pair of stimuli, a transcription of the corresponding text, as well as the question: “Which way of reading the following text do you prefer, A or B?”

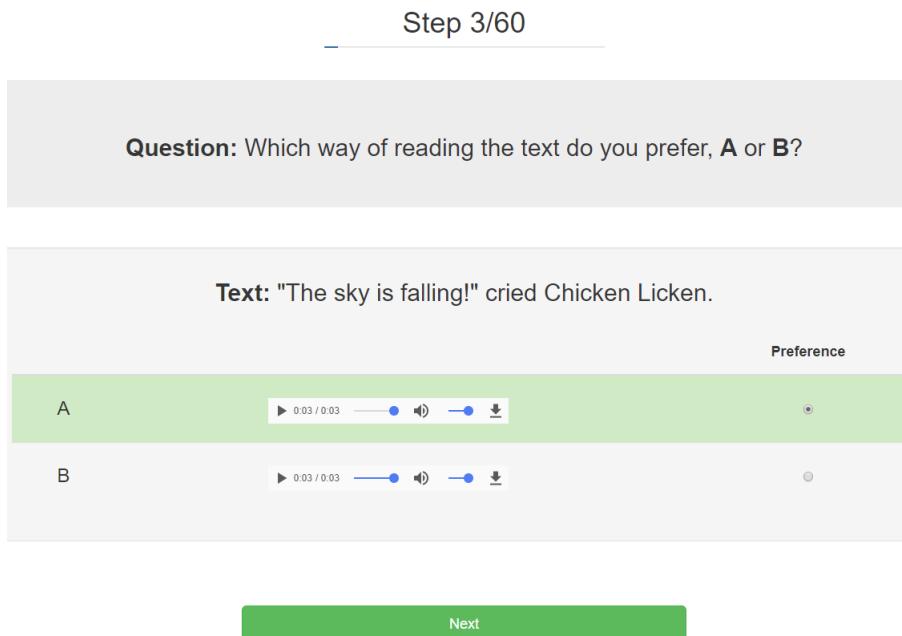


Figure 5.2: Screenshot of one evaluation step.

5.3 Results and Discussion

In total, 40 listeners participated in the evaluation. Of these, 10 were native speakers of English. The results of their evaluation are shown in Figure 5.3 (for more details on the results, see Appendix D).

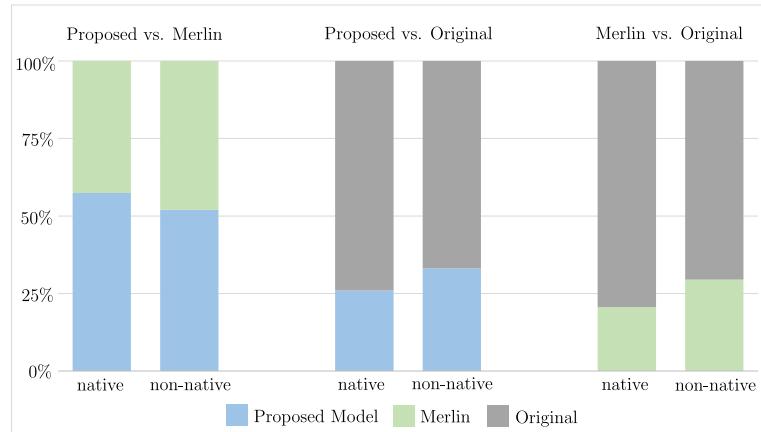


Figure 5.3: Preference evaluation results.

The most noticeable trend is the clear preference of both native and non-native subjects for the natural intonation over the automatically generated ones. This trend is amplified for the native subjects, where preference for the natural intonation can reach as high as 79.4% (Merlin vs. Original).

Comparing the proposed model to Merlin, the two seem to be more or less equivalent for the non-native listeners (52.0% vs. 48.0%). However, if we look at the results for the native subjects, there seems to be a slight preference for the proposed model (57.4% vs. 42.6%).

Overall, these results do not show a very strong preference for either system. However, the difference between the results of the native and non-native subjects is not negligible and might imply that the proposed model is able to capture more specific phenomena, which are going undetected by the non-native subjects. What the nature of these phenomena might be is unclear, but their effect is strong enough to warrant further investigation.

Because the evaluation protocol was designed to be very simple, it also has limitations. For instance, participants had to select their preference without any context. This can be problematic for intonation evaluations, as the same intonation pattern might be correct or incorrect, depending on the context in which it is used. However, the strong preference for the

original shows that subjects were still able to distinguish the systems, even without context.

The evaluation protocol is also limited in its ability to provide more qualitative analysis of the differences between systems. This points to a clear need for a more refined analysis protocol in order to qualify how the proposed model differs from that of Merlin.

Chapter 6

Conclusions

In this thesis, I have presented a dedicated methodology for the modeling of F_0 using a deep learning approach.

The core idea behind the proposed approach is to model the dynamic evolution of the interpolated F_0 through time from a starting position, where the dynamic is parametrized by a sign value for the direction of change, and a quantized magnitude value for the amount of change in such direction. The predicted contour is shifted within the frequency to match the speaker's register. The proposed approach also attempts to inform intonation modeling with semantically richer information by including word embeddings.

After justifying and motivating these underlying ideas, the proposed methodology has been implemented into a DNN model in the context of SPSS. Each part of the implementation design has been justified and motivated.

The implemented intonation model has then been evaluated in comparison to the state-of-the-art parametric TTS system Merlin. The evaluation has shown that the proposed methodology performs just as well as the state-of-the-art and there seems to be a trend for native listeners to actually prefer the proposed model.

Most past and current approaches to intonation modeling have been solely focused on a static description of intonation. This thesis represents a marked departure from these more established approaches, as intonation was modeled as a purely dynamic phenomenon. The evaluation conducted in this thesis has shown that this approach produces results that are on par with (if not slightly better than) the state-of-the-art Merlin system, confirming the validity of the proposed methodology as a new and legitimate direction of research.

6.1 Perspectives and Future Work

6.1.1 Magnitude Relativization

In the proposed F_0 encoding scheme, pitch is encoded as a relative phenomenon: each pitch value is defined in relation to the previous one. This is in opposition the absolute description of static approaches, where each pitch value is defined with respect to its absolute position within the frequency domain.

One possible direction of research could be to explore ways of relativizing the description of pitch even further. In the proposed encoding scheme, magnitude is defined in an absolute way, as each pitch interval is pre-defined based on the underlying reference scale. Future research might try to describe each magnitude value in relation to the previous one.

Describing the magnitude in a relative way would make the representation of pitch more invariant to dilation and compression, as the depth of each fall and rise would be described relative the previous ones. In the proposed methodology, the final predicted contour is shifted to match the speaker's register. If the magnitude is relativized, the final predicted contour could also be adjusted based on a scaling factor to produce a target overall level of compression/dilation. This scaling factor describing compression/dilation could be important to capture differences between speakers, speaking styles, genres, etc.

6.1.2 Larger Speech Corpora

The proposed methodology has been implemented in the context of a deep learning paradigm. DNN training is notoriously data-hungry and only really shines for problems where a vast amount of data is available. However, the evaluation conducted in this thesis was based on a fairly small speech corpus (3 h and 57 mins). In light of the small size of the training corpus, there is only so much that any given DNN model might be able to capture.

An interesting question for future research would be whether the observed trend for native speakers to prefer the proposed methodology would be amplified if the model were trained on a much larger speech corpus. The proposed approach was designed to make use of semantically richer information by means of word embeddings.

However, because of the small size of the corpus, each lemma is observed only a handful of times. Consequently, the DNN model might just be unable to produce a meaningful representation for most lemmata. It would be interesting to explore whether using vastly larger data sets would result

in significant performance gains.

6.1.3 Wavelet Parametrization

Wavelet-based techniques have shown a lot of promise in the field of intonation modeling. However, as a number of researchers are already actively working on it, this thesis was focused on experimenting on new untested ideas. However, this does not entail that the proposed methodology and wavelets are mutually exclusive. In fact, an interesting future direction of research could be to explore ways of extending the proposed methodology with wavelet parametrization.

One of the major issues of the proposed encoding scheme discussed here was its vulnerability to exposure bias at generation time. My explanation for this behavior is the fact that the encoding scheme is based on a chain of prosodic commands, where the interpretation of each is dependent on the previous one. Because of this property, generation errors can easily propagate and accumulate along the generation chain. This issue was largely addressed by using data regularization and data augmentation techniques.

However, wavelets might actually provide a better solution to this problem. By decomposing the contour into sub-components, more global phenomena could be modeled with far fewer data points, which would entail shorter generation chains and, conceivably, less *exposure bias* overall.

6.1.4 Evaluation Protocol

In this thesis, the proposed approach was evaluated with a dedicated evaluation protocol. Because the evaluation was delivered over the internet, the evaluation also had a number of constraints and limitations, which made more qualitative analysis impossible. Despite its limitations, the evaluation protocol was still able to show a strong preference for the natural intonation. This validates the protocol as a preliminary evaluation step, particularly for exploring completely novel and untested ideas.

However, the protocol could not explain the observed trend whereby native subjects seem to prefer the proposed methodology. To better address these questions, it would be important to design a more accurate analysis methodology to expose crucial difference between systems in a more qualitative fashion. A better evaluation protocol for intonation modeling might involve providing more contextual information for the stimuli, recording the subjects' reactions upon listening to stimuli, and asking subjects more detailed questions about why they prefer certain stimuli over others, etc.

Bibliography

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, DOI: 10.1109/72.279181.

Borden, G. J. (1980). Use of feedback in established and developing speech. In *Speech and Language*, volume 3, pages 223–242. Elsevier, DOI: 10.1016/b978-0-12-608603-4.50013-5.

Burnett, T. A., Freedland, M. B., Larson, C. R., and Hain, T. C. (1998). Voice F0 responses to manipulations in pitch feedback. *The Journal of the Acoustical Society of America*, 103(6):3153–3161, DOI: 10.1121/1.423073.

Campbell, N. and Black, A. W. (1997). Prosody and the selection of source units for concatenative synthesis. In *Progress in Speech Synthesis*, pages 279–292. Springer New York, DOI: 10.1007/978-1-4612-1894-4_22.

Chiba, T. and Kajiyama, M. (1941). *The vowel: its nature and structure*. Tokyo-Kaiseikan.

Cho, K., Van Merriënboer, B., Gülcəhre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. URL: <http://aclweb.org/anthology/D/D14/D14-1179.pdf>.

Chung, J., Gülcəhre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, URL: <http://arxiv.org/abs/1412.3555>.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). *CoRR*, abs/1511.07289, URL: <http://arxiv.org/abs/1511.07289>.

Dauphin, Y., de Vries, H., and Bengio, Y. (2015). Equilibrated adaptive learning rates for non-convex optimization. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 1504–1512, Montreal, QC, Canada. URL: <http://papers.nips.cc/paper/5870-equilibrated-adaptive-learning-rates-for-non-convex-optimization>.

Dozat, T. (2016). Incorporating Nesterov momentum into Adam. *ICLR Workshop*, (1):20132016, URL: <https://web.stanford.edu/~tdozat/files/TDozat-CS229-Paper.pdf>.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, URL: <http://dl.acm.org/citation.cfm?id=2021068>.

Espic, F., Valentini-Botinhao, C., and King, S. (2017). Direct modelling of magnitude and phase spectra for statistical parametric speech synthesis. In *Interspeech*, pages 1383–1387, Stockholm, Sweden. URL: http://www.isca-speech.org/archive/Interspeech_2017/abstracts/1647.html.

Fan, Y., Qian, Y., Xie, F., and Soong, F. K. (2014). TTS synthesis with bidirectional LSTM based recurrent neural networks. In *Interspeech*, pages 1964–1968, Singapore. URL: http://www.isca-speech.org/archive/interspeech_2014/i14_1964.html.

Fant, G. (1960). *Acoustic Theory of Speech Production: With Calculations Based on X-ray Studies of Russian Articulations*. The Hague, Netherlands: Mouton.

Fujisaki, H. (1983). Dynamic characteristics of voice fundamental frequency in speech and singing. In *The Production of Speech*, pages 39–55. Springer New York, DOI: 10.1007/978-1-4613-8202-7_3.

Fujisaki, H. (2002). Modeling in the study of tonal feature of speech with application to multilingual speech synthesis. In *Joint International Conference of SNLP Oriental COCOSDA*, pages D1–D9, Hua Hin, Prachuap Khiri Khan, Thailand.

Golik, P., Doetsch, P., and Ney, H. (2013). Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Interspeech*, pages 1756–1760, Lyon, France. URL: http://www.isca-speech.org/archive/interspeech_2013/i13_1756.html.

Greenwood, D. D. (1997). The Mel scale's disqualifying bias and a consistency of pitch-difference equisections in 1956 with equal cochlear distances and equal frequency ratios. *Hearing Research*, 103(1-2):199–224, DOI: 10.1016/s0378-5955(96)00175-x.

Gussenhoven, C., Hart, J. t., Collier, R., and Cohen, A. (1992). A perceptual study of intonation. an experimental-phonetic approach to speech melody. *Language*, 68(3):610–611, DOI: 10.2307/415797.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al. (2014). Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, URL: <http://arxiv.org/abs/1412.5567>.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780, DOI: 10.1162/neco.1997.9.8.1735.

Hoffer, E., Hubara, I., and Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 1729–1739, Long Beach, CA, USA. URL: <http://papers.nips.cc/paper/6770-train-longer-generalize-better-closing-the-generalization-gap-in-large-batch-training-of-neural-networks>.

Hunt, A. J. and Black, A. W. (1996). Unit selection in a concatenative speech synthesis system using a large speech database. In *IEEE Transactions on Audio, Speech, and Language Processing*, volume 1, pages 373–376, Atlanta, GA, USA. DOI: 10.1109/icassp.1996.541110.

Huszár, F. (2015). How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *CoRR*, abs/1511.05101, URL: <http://arxiv.org/abs/1511.05101>.

Kawahara, H., Masuda-Katsuse, I., and de Cheveigné, A. (1999). Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds. *Speech Communication*, 27(3-4):187–207, DOI: 10.1016/s0167-6393(98)00085-5.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, URL: <http://arxiv.org/abs/1412.6980>.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 972–981, Long Beach, CA, USA. URL: <https://papers.nips.cc/paper/6698-self-normalizing-neural-networks>.

Lane, H. and Webste, J. W. (1991). Speech deterioration in postlingually deafened adults. *The Journal of the Acoustical Society of America*, 89(2):859–866, DOI: 10.1121/1.1894647.

Latorre, J. and Akamine, M. (2008). Multilevel parametric-base F0 model for speech synthesis. In *Interspeech*, pages 2274–2277, Brisbane, Australia. URL: http://www.isca-speech.org/archive/interspeech2008/i08_274.html.

Le Maguer, S. and Steiner, I. (2017). Uprooting MaryTTS: Agile processing and voicebuilding. In *Conference on Electronic Speech Signal Processing (ESSV)*, pages 152–159, Saarbrücken, Germany. URL: <http://essv2017.coli.uni-saarland.de/pdfs/LeMaguer.pdf>.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27 (NIPS)*, pages 2177–2185, Montreal, QC, Canada. URL: <https://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization>.

Ling, Z., Deng, L., and Yu, D. (2013). Modeling spectral envelopes using restricted Boltzmann machines and deep belief networks for statistical parametric speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 21:2129–2139, DOI: 10.1109/TASL.2013.2269291.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal. URL: <http://aclweb.org/anthology/D/D15/D15-1166.pdf>.

McAulay, R. and Quatieri, T. F. (1986). Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Audio, Speech, and Language Processing*, 34(4):744–754, DOI: 10.1109/TASSP.1986.1164910.

McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., and Sonderegger, M. (2017). Montreal Forced Aligner: Trainable text-speech alignment using Kaldi. In *Interspeech*, pages 498–502, Stockholm, Sweden. DOI: 10.21437/Interspeech.2017-1386.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 3111–3119, Lake Tahoe, NV, USA. URL: <https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>.

Möbius, B. (2003). Rare events and closed domains: Two delicate concepts in speech synthesis. *International Journal of Speech Technology*, 6(1):57–71, DOI: 10.1023/a:1021052023237.

Morise, M., Yokomori, F., and Ozawa, K. (2016). WORLD: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, 99-D(7):1877–1884, DOI: 10.1587/transinf.2015EDP7457.

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376, URL: <https://ci.nii.ac.jp/naid/20001173129/en/>.

Pierrehumbert, J. (1980). The phonetics and phonology of English intonation. *Unpublished doctoral dissertation, MIT*, URL: <https://dspace.mit.edu/handle/1721.1/16065>.

Pulkki, V. and Karjalainen, M. (2015). *Communication acoustics: an introduction to speech, audio and psychoacoustics*. John Wiley & Sons, ISBN: 978-1-118-86655-9.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732, URL: <http://arxiv.org/abs/1511.06732>.

Ribeiro, M. S. and Clark, R. A. J. (2015). A multi-level representation of F0 using the continuous wavelet transform and the discrete cosine transform. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4909–4913, Brisbane, Australia. DOI: 10.1109/ICASSP.2015.7178904.

Ribeiro, M. S., Watts, O., Yamagishi, J., and Clark, R. A. (2016). Wavelet-based decomposition of F0 as a secondary task for DNN-based speech synthesis with multi-task learning. In *Acoustics, Speech and Signal Processing (ICASSP)*, pages 5525–5529, Shanghai, China. DOI: [10.1109/ICASSP.2016.7472734](https://doi.org/10.1109/ICASSP.2016.7472734).

Ronanki, S., Henter, G. E., Wu, Z., and King, S. (2016). A template-based approach for speech synthesis intonation generation using LSTMs. In *Interspeech*, San Francisco, CA, USA. DOI: [10.21437/interspeech.2016-96](https://doi.org/10.21437/interspeech.2016-96).

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, URL: <http://arxiv.org/abs/1706.05098>.

Sakurai, A., Minematsu, N., and Hirose, K. (2000). Data-driven intonation modeling using a neural network and a command response model. In *Interspeech*, pages 223–226, Beijing, China. URL: http://www.isca-speech.org/archive/icslp2000/i00_3223.html.

van Santen, J. P. H. (1997). Combinatorial issues in text-to-speech synthesis. In *Fifth European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 22–25, Rhodes, Greece. URL: http://www.isca-speech.org/archive/eurospeech1997/e97_2507.html.

van Santen, J. P. H. and Möbius, B. (1997). Modeling pitch accent curves. In *Intonation: Theory, Models and Applications—Proceedings of an ESCA Workshop*, pages 321–324, Athens, Greece. URL: <https://pdfs.semanticscholar.org/a90e/101a59f03c5c65e3ffffe5a1e1e660b44ddc8.pdf>.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, URL: <https://ai.intel.com/wp-content/uploads/sites/53/2017/06/BRNN.pdf>.

Silverman, K., Beckman, M., Pitrelli, J., Ostendorf, M., Wightman, C., Price, P., Pierrehumbert, J., and Hirschberg, J. (1992). TOBI: a standard for labeling English prosody. In *The Second International Conference on Spoken Language Processing, (ICSLP)*, Alberta, Canada. URL: http://www.isca-speech.org/archive/icslp1992/i92_0867.html.

Simard, P. Y., Steinkraus, D., Platt, J. C., et al. (2003). Best practices for convolutional neural networks applied to visual document

analysis. In *7th International Conference on Document Analysis and Recognition (ICDAR)*, pages 958–962, Edinburgh, SCT, UK. DOI: 10.1109/ICDAR.2003.1227801.

Sotelo, J., Mehri, S., Kumar, K., Santos, J. F., Kastner, K., Courville, A., and Bengio, Y. (2017). Char2wav: End-to-end speech synthesis. *ICLR Workshop*, URL: <https://mila.quebec/wp-content/uploads/2017/02/end-end-speech.pdf>.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, URL: <http://jmlr.org/papers/v15/srivastava14a.html>.

Suni, A., Aalto, D., Raitio, T., Alku, P., and Vainio, M. (2013). Wavelets for intonation modeling in HMM speech synthesis. In *Speech Synthesis Workshop (SSW)*, pages 285–290, Barcelona, Spain. URL: http://www.isca-speech.org/archive/ssw8/ssw8_285.html.

Taylor, P. (1994). The rise/fall/connection model of intonation. *Speech Communication*, 15(1-2):169–186, DOI: 10.1016/0167-6393(94)90050-7.

Traber, C. (1990). F0 generation with a database of natural F0 patterns and with a neural network. In *The ESCA Workshop on Speech Synthesis*, pages 141–144, Autrans, France. URL: http://www.isca-speech.org/archive/open/ssw1/ssw1_41.html.

Vainio, M. (2001). *Artificial Neural Network Based Prosody Models for Finnish Text-to-Speech Synthesis*, volume 43 of *Helsingin yliopiston fonetiikan laitoksen julkaisuja*. University of Helsinki, URL: <http://ethesis.helsinki.fi/julkaisut/hum/fonet/vk/vainio/artifici.pdf>.

Vainio, M., Suni, A., Aalto, D., et al. (2013). Continuous wavelet transform for analysis of speech prosody. *Tools and Resources for the Analysis of Speech Prosody*, URL: <http://www.lpl-aix.fr/~trasp/Proceedings/19891-trasp2013.pdf>.

Wang, X., Takaki, S., and Yamagishi, J. (2017a). An RNN-based quantized F0 model with multi-tier feedback links for text-to-speech synthesis. In *Interspeech*, Stockholm, Sweden. URL: <http://www.isca-speech.org/archive/Interspeech2017/abstracts/0246.html>.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., et al. (2017b). Tacotron: Towards end-to-end speech synthesis. In *Interspeech*, pages 4006–4010, Stockholm, Sweden. URL: <http://www.isca-speech.org/archive/Interspeech2017/abstracts/1452.html>.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 4151–4161, Long Beach, CA, USA. URL: <http://papers.nips.cc/paper/7003-the-marginal-value-of-adaptive-gradient-methods-in-machine-learning>.

Wu, Z., Watts, O., and King, S. (2016). Merlin: An open source neural network speech synthesis system. In *Speech Synthesis Workshop (SSW)*, pages 202–207, Sunnyvale, CA, USA. DOI: 10.21437/SSW.2016-33.

Yin, X., Lei, M., Qian, Y., Soong, F. K., He, L., Ling, Z.-H., and Dai, L.-R. (2016). Modeling F0 trajectories in hierarchically structured deep neural networks. *Speech Communication*, 76:82–92, DOI: 10.1016/j.specom.2015.10.007.

Yoshizato, K., Kameoka, H., Saito, D., and Sagayama, S. (2012). Statistical approach to Fujisaki-model parameter estimation from speech signals and its quantitative evaluation. In *Proceedings of the 6th International Conference on Speech Prosody*, volume 1, pages 175–178. URL: <http://slab.hil.t.u-tokyo.ac.jp/publications/download.php?bib=Yoshizato2012SP05.pdf>.

Yu, K., Mairesse, F., and Young, S. (2010). Word-level emphasis modelling in HMM-based speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4238–4241, Dallas, TX, USA. DOI: 10.1109/icassp.2010.5495690.

Yu, K., Toda, T., Gasic, M., Keizer, S., Mairesse, F., Thomson, B., and Young, S. (2009). Probabilistic modelling of F0 in unvoiced regions in HMM based speech synthesis. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3773–3776, Taipei, Taiwan. DOI: 10.1109/ICASSP.2009.4960448.

Yu, K. and Young, S. (2011). Continuous F0 modeling for HMM based statistical parametric speech synthesis. *IEEE Transactions*

on Audio, Speech, and Language Processing, 19(5):1071–1079, DOI: 10.1109/TASL.2010.2076805.

Zatorre, R. J. and Baum, S. R. (2012). Musical melody and speech intonation: Singing a different tune. *PLoS Biology*, 10(7):e1001372, DOI: 10.1371/journal.pbio.1001372.

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. *CoRR*, abs/1212.5701, URL: <http://arxiv.org/abs/1212.5701>.

Zen, H. and Sak, H. (2015). Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4470–4474, South Brisbane, Queensland, Australia. DOI: 10.1109/ICASSP.2015.7178816.

Zen, H., Senior, A., and Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7962–7966, Vancouver, BC, Canada. DOI: 10.1109/ICASSP.2013.6639215.

Appendix A

Intonation Model Label Description

The full set of input and output labels used for the training of the intonation model are reported in table A.1.

Labels	Description	
Syllable Boundary	Size:	3
	Details:	If the sampled point is between two syllables 1, 0 otherwise, <code><unk></code> when unknown.
	Set:	<code><unk></code> , 0, 1
Word Boundary	Size:	3
	Details:	If the sampled point is between two words 1, 0 otherwise, <code><unk></code> when unknown.
	Set:	<code><unk></code> , 0, 1
Syllable stress	Size:	5
	Details:	0 for unstressed, 1 for main stress, 2 for secondary stress, <code><sil></code> for non-speech, , <code><unk></code> when unknown
	Set:	<code><unk></code> , <code><sil></code> , 0, 1, 2
Onset and Rhyme	Size:	4
	Details:	<code><sil></code> for non-speech, , <code><unk></code> when unknown, 0 is the onset of the syllable, R is the rhyme of the syllable, i.e., nucleus+coda
	Set:	<code><unk></code> , <code><sil></code> , 0, R

POS tags	Size:	66
	Details:	<sil> for non-speech, <unk> when unknown. The other labels come from the nltk POS-tagger, and for words such as “don’t” or “I’m” the two POSs predicted by the tagger were spliced together.
Punctuation Before Word	Set:	<unk>, <sil>, NN, DT, VBD, IN, PRP, JJ, RB, NNS, VB, CC, TO, PRP\\$, VBN, VBP, VBG, VBZ, CD, MD, RP, WRB, WP, NNPOS, JJR, PRPVbz, VBDRB, EX, JJS, MDRB, PRPVbz, PRPMD, VBPRB, RBR, NNMD, WDT, PDT, UH, WPVBZ, JJMD, EXVBZ, DTVBZ, VBMD, NNVBP, RBS, VBZRB, WRBPOS, NNP, WDTVbz, FW, JJVBZ, VBPPPOS, VBVBP, RBVBZ, WDTPOS, WP\\$, VBZVBZ, JJPOS, NNSPOS, WRBVBZ, VBPOS, VBPVBP, VBPMD, EXMD, VBZMD, RBPOS
	Size:	55
Punctuation After Word	Details:	<sil> for non-speech, ‘unk’ when unknown. The labels are created by taking whatever punctuation characters are between the current and the previous word. “start” and “end” markers were added at the beginning and end of each utterance. Here the punctuation symbols are separated by the semicolon symbol.
	Set:	<unk> ; <sil> ; - ; , ; start ; .end ; start" ; , " ; . ; !" ; ." ; end ; ."end ; ?" ; !end ; -- ; !"end ; ! ; ?"end ; ?end ; ... ; ...end ; ? ; ' ; " , ; " ; (; : ; ... " ; start... ; ...end ; ,end ;) ; "end ; ?" , ;).end ; :" ; !" , ;) , ; ".end ; start(; start" ... ; ."" ;)end ; .) ; "(; !" . ; ?" .end ; -"end ; start' ; -end ; .!end ; !" , end ; , "end
Punctuation After Word	Size:	Same as “Punctuation Before Word”

	Details:	$\langle \text{sil} \rangle$ for non-speech, ' $\langle \text{unk} \rangle$ ' when unknown. The labels are created by taking whatever punctuation characters are between the current and the subsequent word. "start" and "end" markers were added at the beginning and end of each utterance. Here the punctuation symbols are separated by the semicolon symbol.
	Set:	Same as "Punctuation Before Word"
Lemmata	Size:	See Appendix B
	Details:	See Appendix B
	Set:	See Appendix B
Sign	Size:	3
	Details:	Direction of pitch change
	Set:	-1, 0, 1
Magnitude	Size:	11
	Details:	Amount of pitch change
	Set:	0, 1, 3, 6, 10, 15, 21, 28, 36, 45, 55

Table A.1: Intonation model label description.

Appendix B

Lemmata Label Description

The full set of lemmata labels amounts to 1937 distinct labels. The `<sil>` label is for non-speech, the '`<unk>`' label for unknown words. The lemmata labels come from lemmatized version of the text tokens. To avoid useless frequent words such as character names, we filter out the lemmata that are not found in frequency lists of English based on much larger corpora¹. The full set of lemmata used in the training of the intonation model is the following:

`<unk>`, `<sil>`, the, be, and, of, a, in, to, have, it, i, that, for, you, he, with, on, do, say, this, they, at, but, we, his, from, not, by, she, or, as, an, what, go, their, can, who, get, if, would, her, all, my, make, about, know, will, up, one, time, there, year, so, think, i'm, when, don't, which, them, some, didn't, me, people, take, out, into, just, see, him, your, come, could, now, it's, than, like, other, how, then, its, our, two, more, these, want, way, look, first, also, new, because, day, use, no, man, find, here, thing, give, many, well, only, tell, very, even, back, any, good, woman, through, us, life, child, work, down, may, after, should, call, world, over, school, still, wasn't, try, last, ask, you're, need, too, feel, three, state, never, become, between, high, really, something, most, another, much, that's, family, own, leave, put, old, while, mean, keep, student, why, let, great, same, big, group, begin, seem, country, help, talk, where, i'll, turn, problem, every, start, hand, might, american, couldn't, show, part, against, place, such, again, few, case, week, company, each, can't, right, hear, question, during, play, run, small, number, off, always, move,

¹The frequency list used to filter out words is derived from the The British National Corpus (<http://www.natcorp.ox.ac.uk/using/index.xml?ID=freq>).

night, live, mr, point, believe, hold, today, bring, happen, next, god, without, before, large, million, must, home, he's, under, water, room, write, mother, area, national, money, story, young, fact, month, different, lot, book, eye, i'd, job, word, though, business, side, kind, four, head, far, black, long, both, little, house, wouldn't, yes, since, i've, around, friend, important, father, sit, away, until, power, hour, game, often, yet, line, end, among, ever, stand, bad, lose, member, pay, law, meet, car, city, almost, include, continue, set, later, name, five, once, white, least, learn, real, change, team, minute, best, she's, several, he'd, idea, body, nothing, ago, lead, understand, watch, together, hadn't, follow, parent, stop, face, sure, already, speak, others, read, allow, add, office, spend, door, person, create, war, doesn't, history, party, within, grow, open, morning, walk, low, win, girl, what's, early, food, moment, himself, air, teacher, force, offer, we're, enough, across, although, remember, foot, second, boy, anything, maybe, able, everything, love, music, appear, buy, probably, human, wait, serve, market, die, send, expect, sense, build, stay, fall, oh, nation, plan, cut, college, death, course, someone, she'd, experience, behind, reach, local, kill, six, remain, won't, suggest, class, raise, care, perhaps, there's, late, hard, field, else, pass, sell, sometimes, along, isn't, themselves, report, better, decide, strong, possible, heart, leader, light, voice, wife, whole, police, mind, finally, pull, return, free, price, less, explain, son, hope, develop, position, carry, town, road, drive, arm, true, break, thank, international, building, action, full, join, further, director, view, player, agree, especially, record, pick, wear, paper, special, space, ground, form, support, event, they're, whose, matter, everyone, you've, couple, hit, base, star, table, court, produce, eat, teach, half, easy, cost, figure, street, itself, either, cover, quite, picture, you'll, clear, piece, land, doctor, wall, worker, news, movie, certain, north, simply, third, catch, step, baby, type, attention, draw, goodbye, tree, we'll, red, nearly, choose, cause, hair, century, window, listen, soon, chance, brother, period, summer, realize, hundred, plant, short, letter, choice, single, rule, daughter, south, husband, floor, haven't, church, close, thousand, fire, future, wrong, involve, anyone, weren't, bank, myself, sport, board, officer, rest, performance, fight, throw, top, quickly, past, goal, bed, order, author, fill, drop, blood, upon, aren't, push, store, reduce, sound, fine, near, page, enter, share, poor, race, similar, hot, usually, dead, rise, animal, shoot, east, save, seven, despite, eight, thus, happy, exactly, protect, approach,

determine, size, dog, serious, ready, sign, thought, answer, you'd, mile, left, lie, prepare, giggle, whatever, success, argue, cup, stuck, character, recognize, wonder, attack, herself, television, box, hung, training, pretty, everybody, lay, general, feeling, bore, message, outside, arrive, forward, present, skill, sister, chuckle, stage, ok, they'd, miss, sort, act, shouldn't, ten, mumble, station, blue, indeed, truth, song, check, leg, dark, rather, laugh, guess, prove, hang, entire, rock, forget, claim, enjoy, cold, final, main, green, card, above, seat, amaze, nice, trial, expert, spring, visit, imagine, tonight, huge, ball, finish, yourself, charge, popular, onto, fly, weapon, we've, pain, wide, shake, direction, chair, fish, camera, perform, sadness, bit, suddenly, discover, production, treat, trip, evening, inside, adult, worry, deep, edge, let's, writer, trouble, throughout, challenge, fear, shoulder, middle, sea, dream, beautiful, instead, improve, stuff, somebody, hotel, soldier, heavy, bag, heat, marriage, tough, sing, pattern, skin, owner, ahead, who's, yard, beat, finger, garden, notice, collection, partner, kitchen, shot, we'd, wish, safe, demon, mouth, victim, newspaper, smile, score, audience, rich, dinner, travel, none, front, born, wind, key, fast, alone, bird, speech, southern, eventually, forest, global, restaurant, judge, customer, corner, swirl, railway, version, safety, troop, numb, hurt, track, strike, sky, nobody, powerful, perfect, nine, announce, touch, please, completely, sleep, it'll, replace, british, camp, brain, date, battle, afternoon, african, sorry, fan, stick, easily, hole, growl, chinese, ship, solution, stone, slowly, scale, university, introduce, driver, he'll, park, spot, thunder, boat, drink, sun, distance, wood, handle, mountain, winter, village, refuse, roll, gain, hide, gold, club, farm, shape, crowd, nervously, strength, band, horse, prison, ride, guard, demand, deliver, wild, observe, advantage, hiss, quick, pound, bright, guest, tiny, block, protection, sting, settle, hmm, feed, collect, scowl, mostly, lesson, river, snort, count, marry, tomorrow, path, ear, shop, folk, lift, competition, jump, gather, sob, fit, cry, warm, insist, christmas, spread, soft, egg, murder, engage, tug, coffee, speed, cross, anyway, saturday, female, wave, afraid, she'll, quarter, native, wonderful, suit, blow, destroy, adore, cook, burn, shoe, hey, mistake, flutter, fake, clothes, quiet, dress, cool, bone, chief, yawn, below, promise, they'll, bus, dangerous, remind, moral, united, victory, following, dread, medicine, tour, photo, grab, hop, fair, pair, famous, exercise, knee, flower, hire, actor, birth, search, tie, circle, bottom, island, train, lady, neck, damage, plastic,

tall, plate, male, alive, football, chicken, army, claw, shut, map, extra, danger, welcome, hasn't, rain, nod, leaf, dry, russian, pool, thorn, climb, sweet, engine, fourth, salt, metal, fat, ticket, lip, dove, strange, disappear, lunch, moonlight, somewhere, farmer, sugar, mock, explore, grumble, enemy, planet, twirl, invite, repeat, hum, howl, carefully, married, weather, monday, bear, pocket, shiny, squeal, surprise, pierce, breath, sight, crumble, straight, belong, okay, photograph, empty, trail, somehow, organize, storm, thanks, expensive, yellow, shadow, dance, ring, mark, bridge, sunday, rainbow, anymore, thinking, tickle, visitor, angry, crew, accident, meal, capture, glide, prefer, earth, chest, thick, cash, beauty, salty, link, root, nose, declare, daddy, bottle, america, hardly, sick, defend, sheet, mix, foul, slow, wake, brown, shirt, warn, petal, cat, ponder, guide, snow, english, steal, they've, slip, meat, funny, soil, how's, blame, due, dart, crazy, chain, branch, relief, dad, fright, kick, ancient, fee, hurl, golden, german, silence, bowl, bound, except, trickle, hall, trust, row, afford, meanwhile, fix, coward, creak, dusty, bedroom, secret, nurse, ache, opposition, anywhere, master, puff, everywhere, wing, lord, pour, stir, unseen, terrible, gulp, grant, hero, cloud, stretch, winner, pepper, bark, busy, tip, aim, shudder, vegetable, dish, fun, afterwards, opening, tear, grimace, whistle, league, hat, rush, tired, fry, luckily, apart, match, barely, beneath, beside, proud, peek, enormous, wheel, narrow, cream, gate, solid, hill, noise, grass, careful, celebrate, useful, crown, taste, milk, escape, cast, inch, closely, convince, height, unusual, plenty, sharp, explanation, roof, weak, signal, forever, association, twenty, knock, warning, cheese, sir, bread, scream, excellent, deeply, lucky, drag, guilty, arrest, wash, sad, post, steel, shout, violent, silent, suppose, tea, joke, description, slide, wedding, opponent, lake, bend, shall, sand, tale, arrange, reply, opposite, prince, lock, deserve, stream, sale, pot, grand, mine, hello, spanish, knife, countryside, coat, potato, urge, dust, breathe, ordinary, rarely, pack, numerous, iron, passion, priest, amazing, advance, shock, kiss, cap, juice, whenever, boss, king, boot, asian, bean, creature, usual, round, breakfast, luck, smell, nervous, toss, bury, pray, journey, surely, tower, smoke, glance, toy, prisoner, nearby, birthday, castle, perfectly, coast, silver, flag, whisper, gentleman, moon, swing, dig, wet, pan, mayor, pink, poem, grandmother, preparation, wooden, cricket, concert, jail, giant, cake, pop, quietly, shell, onion, brand, phrase, snap, hip, killer, gang, heaven, rough, yell, clock, chocolate, sweep, button, bell,

darkness, clothing, fence, react, furniture, cheek, pant, stranger, broken, apple, electric, bet, stupid, fortune, shopping, cousin, wive, slave, dirt, odd, originally, bullet, tight, chart, square, gently, sensible, strip, friendly, deck, tournament, pride, bake, freeze, platform, sink, overseas, thirty, crash, tap, swim, tire, fault, loose, rice, rugby, stair, proof, adventure, tongue, shelter, rub, entrance, fade, net, funeral, clever, squeeze, mask, stable, pretend, steady, oven, nowhere, exciting, ill, adapt, honey, pale, musician, flee, carriage, comfort, scared, plot, gesture, chapter, shade, tail, custom, fifteen, soup, celebration, pile, closer, besides, meter, incredible, fighter, fifty, rid, cow, trick, federation, duke, qualify, asleep, barrel, medieval, bite, loud, glove, delay, stroke, badly, murmur, urgent, cotton, float, orange, blade, cabin, desperate, yours, pitch, brilliant, boom, hungry, penny, wander, shrug, flame, collapse, comedy, twelve, brush, wise, running, basket, ah, fighting, ugly, worried, ghost, magnificent, cooking, ban, awful, heading, lovely, specially, tactic, blanket, mouse, chase, brick, cupboard, horror, recording, pie, gaze, courage, swear, defeat, slice, dear, coal, uncle, captain, sigh, sadly, dare, soccer, tunnel, toe, abroad, mess, shine, upset, reward, gentle, log, invent, laughter, insect, interrupt, magic, hunt, lightly, excited, dull, flour, bitter, bare, candy, pity, pleased, beg, slam, melt, midnight, greet, corridor, march, snake, excuse, pig, classical, flash, duck, roman, confuse, excitement, plead, downstairs, trunk, swallow, fetch, trap, princess, inspector, plain, burst, cave, monster, fancy, obstacle, rip, herb, delighted, flood, pen, nightmare, arena, forgive, striker, drift, drain, warrior, mud, hurry, temple, suck, broadcast, leap, pond, guilt, skirt, tune, railroad, horn, strain, nonsense, pad, bat, bye, creep, clash, grateful, grip, supper, wherever, forty, trait, turkey, reserve, beam, stitch, smash, shield, thumb, horrible, ruler, twist, relieve, rebel, forehead, bounce, hook, inn, fox, needle, scare, ankle, rescue, firmly, detective, rider, noon, poster, crawl, handsome, sum, halt, hug, punish, doorway, happiness, bearing, emperor, bath, purple, reluctant, thief, eating, stamp, saint, brass, sharply, yacht, fossil, peel, lump, goodness, referee, dreadful, candle, hut, servant, vanish, summon, polish, chop, silk, popularity, fling, scary, trophy, angel, rage, precious, hidden, stumble, lonely, dawn, silly, tide, kit, wicket, seal, gardener, fool, rear, softly, burning, arouse, useless, tremble, cart, obey, chat, knight, o'clock, weep, treasure, mist, deed, caravan, grief, rocket, tackle, bow, ours, furious, bubble, barn, sword, tightly, protective, tuck, faint, queen,

sail, stadium, bloody, nest, lane, steam, cage, stag, clutch, temper, wolf, throne, grin, bug, bless, aunt, dive, mixed, grasp, calm, haul, curl, ruin, silently, sunshine, bang, bush, polite, brow, miserable, terrify, sheep, cab, teammate, stride, snatch, bee, loop, shiver, whip, fury, boil, murderer, monk, hammer, despair, spectator, sock, eleven, luxury, maid, gasp, enclose, balcony, sailor, surrender, grim, rolling, spell, helmet, lion, glare, royal, panic, wicked, cliff, ashamed, scramble, flick, angrily, amuse, torch, delicious, dragon, exclaim, delightful, soap, hatred, noisy, punch, staircase, passing, flourish, purse, shed, elephant, worm, cheat, triangle, fever, rabbit, coin, stain, loudly, upstairs, shatter, bored, dough, stool, sniff, foolish, stab, rude, flock, poison, moor, oak, eighth, herd, tiger, stagger, toast, tease, roar, fairy, ray, scent, sack, nasty, sleeping, nun, heap, fierce, rob, worse, passport, unfair, chunk, frog, disguise, courtyard, ladder, jungle, invade, sip, skip, dip, feather, boast, villager, clearing, glow, weed, kindly, attacker, chimney, witch, lone, sneak, monkey, lick, disturb, trench, scar, kite, gloom, tumble, cure, hunger, haunt, faster, gossip, spare, halfway, hen, boiler, carrot, cling, blink, procession, plunge, vicious, steer, cheer, slump, chew, magical, mole, feast, majesty, goalkeeper, scratch, awake, groan, craftsman, meadow, lid, sprinkle, voyage, jewel, goat, bump, banana, palace, fuss, ram, rotten, hover, beard, brake, moan, fur, brutal, happily, soar, unhappy, straw, exhaust, globe, blast, overnight, fare, screw, warmth, cruel, mansion, cottage, balloon, fantastic, frown, ha, marble, defender, mutter, arrow, meantime, spy, brave, importantly, sunny, straighten, delight, spine, fog, butterfly, kneel, scissors, sharpen, saw, aboard, sow, sweeten, astonish, bravery, oar, merry, berry, fatten, greed, conquer, messenger, tame, paw, liar, ripe, donkey, pearl, handkerchief

Appendix C

Segmental Synthesizer Label Description

Labels	Description	
Current Phone	Size:	45
	Details:	The phone currently observed. The phone set is based on the OALD ¹ , augmented with an extra symbol for silence.
	Set:	ə, @®, a, aa, ai, au, b, ch, d, dh, e, e®, ei, f, g, h, i, i®, ii, jh, k, l, m, n, ng, o, oi, oo, ou, p, r, s, sh, sil, t, th, u, u®, uh, uu, v, w, y, z, zh
Previous Phone	Size:	Same as “Current Phone”
	Details:	The phone before the currently observed one.
	Set:	Same as “Current Phone”
Before Previous Phone	Size:	Same as “Current Phone”
	Details:	The phone before the previously observed one.
	Set:	Same as “Current Phone”
Next Phone	Size:	Same as “Current Phone”
	Details:	The phone after the currently observed one.
	Set:	Same as “Current Phone”

¹ http://www.cstr.ed.ac.uk/downloads/festival/2.4/festlex_OALD.tar.gz

After Next Phone	Size:	Same as “Current Phone”
	Details:	The phone after the subsequent phone.
	Set:	Same as “Current Phone”
Phone Duration	Size:	1
	Details:	Number of frames for each phone divided by 100 to make it fit approx. within a 0-1 range.
	Set:	n/a
Log₂(F₀)	Size:	1
	Details:	Interpolated $\log_2(F_0)$, divided by 10 to make it fit approx. within 0-1 range.
	Set:	n/a
VUV	Size:	2
	Details:	1 for voiced, 0 for otherwise.
	Set:	0, 1
BAP	Size:	6
	Details:	5 BAP coefficients and corresponding gain
	Set:	n/a
MGC	Size:	61
	Details:	60 MGC coefficients and corresponding gain
	Set:	n/a

Table C.1: Segmental synthesizer label description.

Appendix D

Evaluation Results

	P. vs. M.	P. vs. O.	M. vs. O.
Proposed	93	41	
Merlin	69		34
Original		117	131

Table D.1: Native speakers evaluation results.

	P. vs. M.	P. vs. O.	M. vs. O.
Proposed	302	196	
Merlin	279		172
Original		395	411

Table D.2: Non-native speakers evaluation results.

	P. vs. M.	P. vs. O.	M. vs. O.
Proposed	57.4%	25.9%	
Merlin	42.6%		20.6%
Original		74.1%	79.4%

Table D.3: Native speakers evaluation results (in percentage).

	P. vs. M.	P. vs. O.	M. vs. O.
Proposed	52%	33.2%	
Merlin	48%		29.5%
Original		66.8%	70.5%

Table D.4: Non-native speakers evaluation results (in percentage).

Appendix E

The Implementation

The code is implemented in Python 3 and is organized into four repositories: two for the intonation model (one for training and one for inference) and two for the segmental synthesizer (one for training and one for inference). The choice of splitting the project into smaller sub-projects was made for the sake of modularity. For instance, some users might only be interested in the intonation model, whereas others might only be interested in the implementation of the segmental synthesizer to carry out their research on prosody. All four sub-projects are built by means of open-source build automation system Gradle¹.

E.1 Intonation Model Implementation

The implementation of the intonation model is hosted at the following links: https://github.com/ftomb/intonation_model_training and https://github.com/ftomb/intonation_model_inference. The first link is for the training of the model, the second to run inferences.

To run the implementation, you will first need the following external installed tools:

- Java
- SoX²

In addition, you will need the following Python packages:

- tgt
- numpy

¹<https://gradle.org/>

²<http://sox.sourceforge.net/>

- `nltk`³
- `scipy`
- `tensorflow`⁴ (1.2 and above)
- `pyworld`
- `pysptk`

To train the model, you need to place the following three folders into the (`/src`) directory of the first repository (https://github.com/ftomb/intonation_model_training):

- `txt` (text files)
- `wav` (wave files)
- `textgrid` (textgrid files)

The textgrid files must contain two tiers: one for the words and one for the phones. The name of the word tier must contain the string “words” and, likewise, the name of the phone tier must contain the string “phones”. If you strip the phones of the stress information for the alignment phase, make sure you add the stress information back into the phone tier.

For the syllabifier to work, the phones in the textgrids must be based on the phone set used in the OALD⁵ phone set. It is also possible to use other phone sets such as the one the Carnegie Mellon University (CMU) Pronunciation Dictionary is based on.

To use the CMU set, you will have to modify the `phone_set_path` variable in the `build.gradle` script to point to the CMU file provided in the `/src` directory. To use different phone sets or difference languages, you will have to place a `json` file similar to the ones provided in the `/src` directory, in which you provide the consonant, vowel and onset sets of the language.

Next, you have to provide the number of epochs you want the model to train for. The value can be adjusted by modifying the `n_epochs` variable in the `build.gradle` script. For my implementation (3h and 57mins of speech), 20 epochs were sufficient.

To start training the model, run the following command, where `[number_epochs]` is replaced by the number of epochs you want to train the model for:

³In particular you need to make sure have the *wordNetLemmatizer* installed (<http://www.nltk.org/modules/nltk/stem/wordnet.html>), as well as the NLTK *WordNet* corpus downloaded

⁴<https://www.tensorflow.org/>

⁵ http://www.cstr.ed.ac.uk/downloads/festival/2.4/festlex_OALD.tar.gz

```
./gradlew synthesize_wav_[number_epochs]
```

This command will automatically train a model for the specified number of epochs. At each epoch a model will be saved. The model is marked by a number that corresponds to the epoch it was produced at.

Then, for each trained model, the command will use a vocoder to synthesize the entire validation set using the acoustic features from the original recordings. As the synthesis of segmental features by means of DNN is computationally expensive, for this validation stage the vocoder is used instead. After the synthesis process is complete, you can listen to waveforms produced at each epoch, so that you can pick the model you deem most accurate. Each waveform is marked at the end by the model number it was synthesized from. The wave files will be generated inside the `/build/33_synth_wav` folder

Finally, you can collect the files you will need at inference time. These include:

- `/build/18_NN_dictionaries/sign_label_dictionary.json`
- `/build/18_NN_dictionaries/magn_label_dictionary.json`
- `/build/20_merged_dictionaries/inference_dictionaries.json`
- `/build/28_frozen_models/frozen_model_[chosen_number]`

To run the inference, you need to place these files into a folder named `model`. You also need to rename the `frozen_model_[chosen_number]` to `frozen_model` (i.e., remove the epoch number marking it at the end). The `model` folder must then be placed inside the (`/src`) directory of the second repository (https://github.com/ftomb/intonation_model_inference). In addition you have to place the following two folders with their corresponding files inside the `/src` folder:

- `txt` (text files)
- `textgrid` (textgrid files)

Similarly to the training process, if you used a different phone set, you need to modify the `phone_set_path` variable in the `build.gradle` script to point to the file provided in the `/src` directory.

To start the inference, run the following command:

```
./gradlew convert_to_hertz
```

This command will automatically generate F_0 contours for the provided data. The F_0 files will be generated inside the `/build/32_synth_f0s` folder.

E.2 Segmental Synthesizer Implementation

The implementation of the intonation model is hosted at the following links: https://github.com/ftomb/segmentalsynthesizer_training and https://github.com/ftomb/segmentalsynthesizer_inference. The first link is for the training of the model, the second to run inferences.

To run the implementation, you will first need the following external installed tools:

- Java
- SoX⁶

In addition, you will need the following Python packages:

- tgt
- numpy
- tensorflow⁷ (1.2 and above)
- pyworld
- pysptk

To train the model, you need to place the following three folders into the (`/src`) directory of the first repository (https://github.com/ftomb/segmentalsynthesizer_training):

- wav (wave files)
- textgrid (textgrid files)

The textgrid files must contain phone tier. The name of the phone tier must contain the string “phones”.

Next, you have to provide the number of epochs you want the model to train for. The value can be adjusted by modifying the `n_epochs` variable in the `build.gradle` script. For my implementation (3h and 57mins of speech), 25 epochs were sufficient.

To start training the model, run the following command, where `[number_epochs]` is replaced by the number of epochs you want to train the model for:

```
./gradlew synthesize_[number_epochs]
```

⁶<http://sox.sourceforge.net/>

⁷<https://www.tensorflow.org/>

This command will automatically train a model for the specified number of epochs. At each epoch a model will be saved. The model is marked by a number that corresponds to the epoch it was produced at.

Then, for each trained model, the command will use the DNN model to synthesize the entire validation set. After the synthesis process is complete, you can listen to waveforms produced at each epoch, so that you can pick the model you deem most accurate. Each waveform is marked at the end by the model number it was synthesized from.

Finally, you can collect the files you will need at inference time. These include:

- `/build/05_phone_dictionary/phone_dictionary.dict`
- `/build/07_input_mean_std/input_mean_std.json`
- `/build/08_output_mean_std/output_mean_std.json`
- `/build/13_frozen_models/frozen_model_[chosen_number]`

To run the inference, you need to place these files into a folder named `model`. You also need to rename the `frozen_model_[chosen_number]` to `frozen_model` (i.e., remove the epoch number marking it at the end). The `model` folder must then be placed inside the (`/src`) directory of the second repository (https://github.com/ftomb/segmental_synth_inference). In addition you have to place the following folder inside `/src`:

- `textgrid` (textgrid files)

To start the inference, run the following command:

```
./gradlew synthesize
```

This command will automatically generate wave files for the provided data. The wave files will be generated inside the `/build/wav` folder.

