Contents

List of Figures

Chapter 1

Introduction

Information regarding the initial work-plan, deviations from it, work packages, gantt diagram...

Chapter 2

State of the art

This chapter gives a brief introduction to the background of this project. The first thing that is discussed is classical methods of speech synthesis, as well as a brief mention of deep learning methods used in this particular task.

Next there is a brief introduction to Deep Learning and deep architectures (both definition and training) a field that is applied to this project. By the end of this chapter, the necessary background will have been introduced for its application in the next chapter.

2.1 Speech synthesis

Speech synthesis is the process of generating a synthetic speech signal, emulating that of a real human being. More specifically, this project focuses on synthesizing the signal given text level information. That means, mapping a piece of text to the sounds a human would make to utter said text, and it has to be possible to generate any text that is given to the system.

Speech synthesis has many applications, from medical to recreational. These systems can be used to help speech impaired people and it can also be used to design more human-like interactions objects.

2.1.1 Unit selection

Unit selection systems render a speech signal by concatenating waveform fragments from a large single-speaker database [12]. The outcome of this technique is highly natural-sounding speech since the fragments that are concatenated come from real waveforms. However, while this system is good at producing natural sounding speech, it lacks the flexibility to produce new speaker sounds or vary the speaking style, as described in [13], since we are limited by the waveforms database. Each unit also has an associated feature vector containing prosodic information such as pitch and formants.

To choose the best sequence of concatenated units, a cost of concatenation is defined [**?**] given a sequence of units $u_1^N = \{u_1, u_2, ..., u_N\}$ and their corresponding prosodic vectors p_1^N:

$$C_T^p = C_T^p(u_1^N) = \sum_{i=1}^{N} C_p(p_t, p_i) \tag{2.1}$$

$$C_T^c = C_T^c(u_1^N) = \sum_{i=2}^{N} C_c(u_{i-1}, u_i) \tag{2.2}$$

Where C_p is a function that measures the cost of using a specific feature vector (for example if the units have matching formants and pitch) to match a given target p_t and C_c is the cost of concatenation of two consecutive units (for example a unit ends with the same pitch level that the next one starts with). Finding the best sequence of units is done by using the Viterbi algorithm to minimize the total concatenation cost:

$$u_1^N = \underset{u_1^N}{argmax} \left\{ \rho \cdot C_T^p(u_1^N) + (1-\rho) \cdot C_T^c(u_1^N) \right\} \tag{2.3}$$

$$0 \leq \rho \leq 1 \tag{2.4}$$

Where ρ is a weighting term that can be used to pay more attention to the targets that to the concatenation, or vice versa.

2.1.2 Statistical Parametric Speech Synthesis

In statistical-parametric speech synthesis (SPSS) a set of spectral and excitation parameters are extracted from a speech corpus and then a statistical generative model (typically a hidden Markov model or HMM) is trained to model the them for use in speech synthesis.

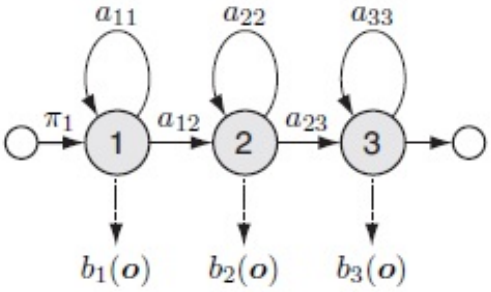

Figure 2.1: A 3-state left-to-right Hidden Markov Model.

As seen in Figure 2.1, a HMM is a finite state machine that can be described by the tuple $\lambda = \langle A, B, \Pi \rangle$ where A are the state transition probabilities, B are the output distributions on each state and Π are the initial state probabilities (spectral & excitation output parameters). HMMs are used as generative model since they provide a time-discrete series of observations from a distribution of the speech parameters. In contrast with unit selection, HMM based speech synthesis models offer a more compact representation than unit selection since they don't require a large database after the models have been trained.

Moreover, it also overcomes the limitation of only being limited to one speaker, since we can easily modify the speaker characteristics and speaking styles by transforming the HMM parameters appropriately as describes in [19].

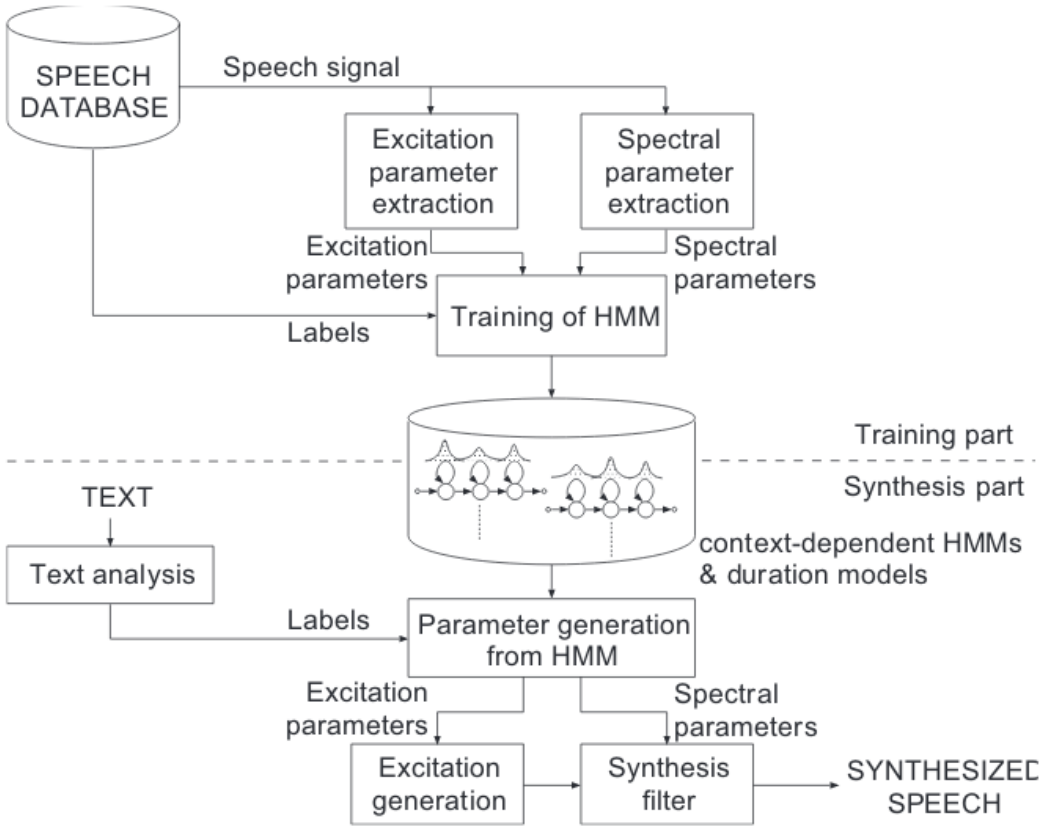

Figure 2.2: SPSS system based on HMMs. This scheme corresponds to the HTS framework [19].

2.1.3 Recurrent Neural Network-based Speech Synthesis

Another approach to speech synthesis more closely related to the implementation of this project, is using recurrent neural networks (RNNs) to process a sequence of linguistic features to predict the same excitation and spectral parameters from SPSS.

RNNs are covered in more detail in a later section of this chapter. As describes by [6], this scheme consists of using neural networks to generate parametric speech in two stages: a duration model that predicts the duration of a phoneme in a sentence, and an acoustic model to predict the spectral and excitation parameters for as many time steps as predicted by the duration model. Figure 2.3 shows the complete scheme of this approach.

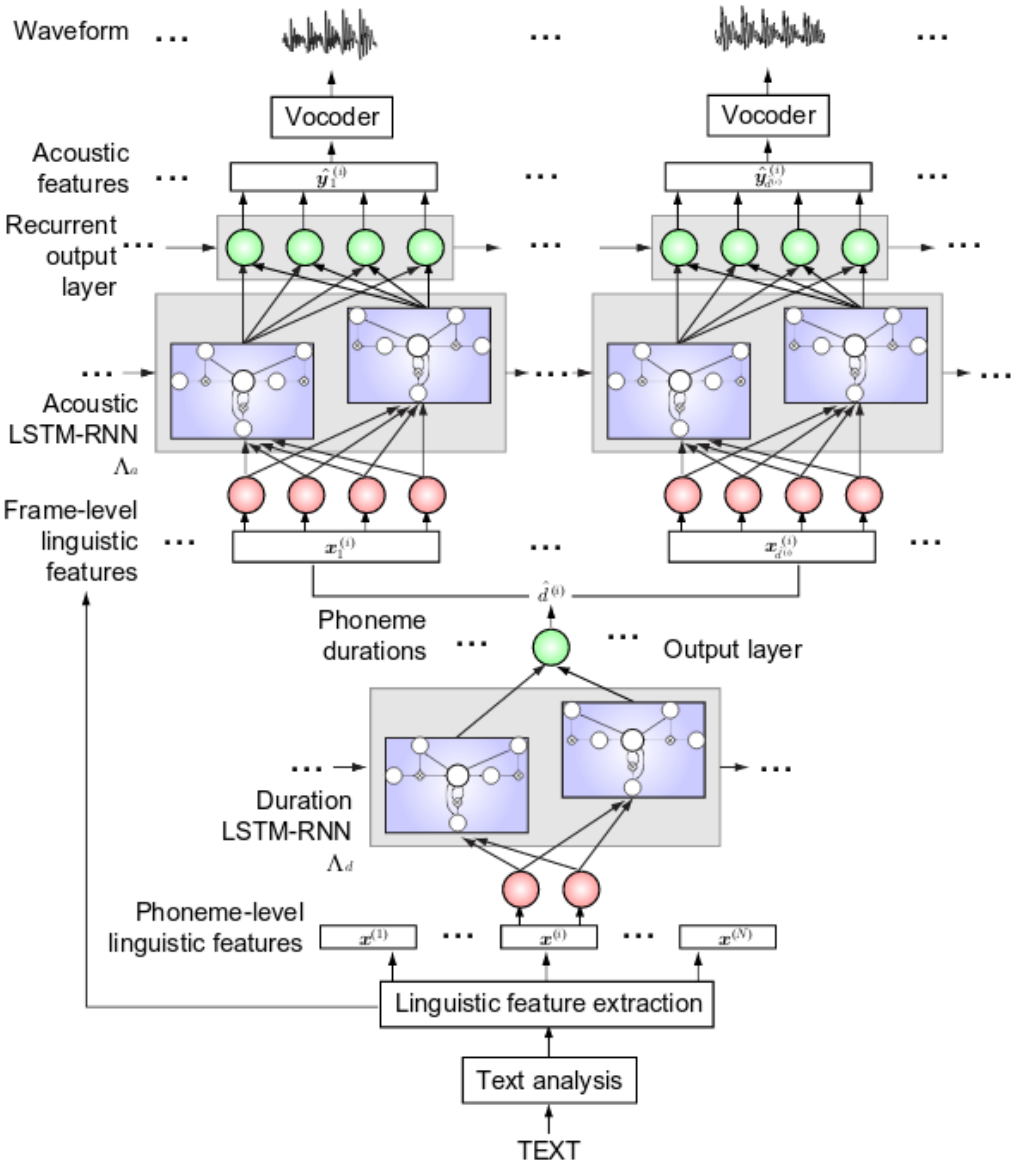

Figure 2.3: RNN-based SPSS scheme. Concepts such as LSTM that appear in the figure are covered at later section of this chapter.

2.1.4 Expressive

Summary of the bibliography of the approaches to expressive speech synthesis.

2.2 Text classification for sentiment Analysis

2.3 Deep Learning

In recent years, deep structured learning, or more commonly called deep learning is a set of techniques that has risen in popularity and established itself as a new area of machine learning among the scientific and research communities. [9]

[1] discussed the limitations of shallow architectures and provided a theoretical basis to the advantages of deeper ones. But it has only been in recent years that such architectures have been able to be put into production with ever-growing popularity thanks to compute technologies being every time more available and affordable to consumers.

With deep learning architectures, it is possible to learn complex non-linear representations from large amounts of data. Raw data can be processed and turned into several levels of higher level representations to make it possible to reason about them and make tasks such as classification easier and more effective.

In addition to more compute power, large datasets that can take advantage of deep models such Imagenet [16] are being made publicly available. These datasets are often used as benchmarks and serve as points of reference to drive research and development forward.

In summary, one could describe Deep Learning as models capable of obtaining high level information from low level one, and the compute resources and databases that make training of these models possible.

2.3.1 Deep Neural Networks

In Deep Learning, a common architecture is the so called Deep Neural Network (DNN), which is an artificial neural network with a large number of hidden layers. The desire to decomposing a signal into higher levels of abstraction drives such neural networks configurations, and since more layers require more trainable parameters, large amounts of training data are also needed to train them.

DNNs are defined by stacking several layers of basic units called neurons. In each layer, a linear operation takes place, where the inputs $\mathbf{x} = \{1, x_1, x_2, ..., x_N\}$ (1 is for a bias term) are linearly combined by a set of weights that are characteristic of each layer. The output of this linear operation is fed to a non-linear activation function (such as a sigmoid (2.6) or a tanh) which introduces the non-linearities of the system and is differentiiable (property that will be relevant for optimizing or training the weights). Equation 2.5 shows the operation

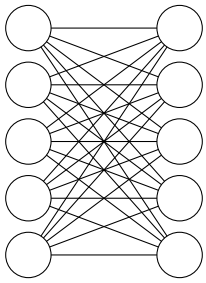

Figure 2.4: Neural network basic configuration. Each layer consists of a net of nodes chich perform a linear operation of the inputs, before being fed to an activation function at the output and fed to the next layer.

that takes place in the i-th layer, where $\mathbf{W_i}$ is the matrix of weights, $\mathbf{x_i}$ is the input vector of the layer, $\mathbf{y_i}$ is the output vector and σ is the activation function.

$$\mathbf{y_i} = \sigma\left(\mathbf{W_i} \cdot \mathbf{x_i}\right) \tag{2.5}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

The outputs of this activation function are fed to the next layer of neurons until the last layer of a network, typically a softmax activation function for classification tasks or linear operation in regression ones. Figure 2.4 shows a neural network configuration.

2.3.2 Optimization

Optimizing, or training, a DNN is the process of finding the best possible set of weights that accomplish the best results in a given task. A task that DNNs are used for is classification, where a label is assigned to each input vector (predicting the wether an image is from a cat or a dog) and also regression (predicting the expected cost of a property given the land size and proximity to the coast).

$$L(\mathbf{x}; \mathbf{w}) = \sum_{\mathbf{x}_k} L_i(\mathbf{x_k}) \tag{2.7}$$

$$w_i[n+1] = w_i[n] - \lambda \frac{\partial L(\mathbf{x}; \mathbf{w}[n])}{\partial w_i} \tag{2.8}$$

To optimize the network a cost or loss function is defined to measure the error of the predictions (such as the root mean square error or the cross-entropy error [10]). given a loss, we can find improve the performance of a model by translating the weight vector of the model in the direction of the gradient.

Equation 2.7 is a general loss function, computed by adding the errors of a batch of vectors of a dataset. These vectors are the examples from where the network learns. The size of the

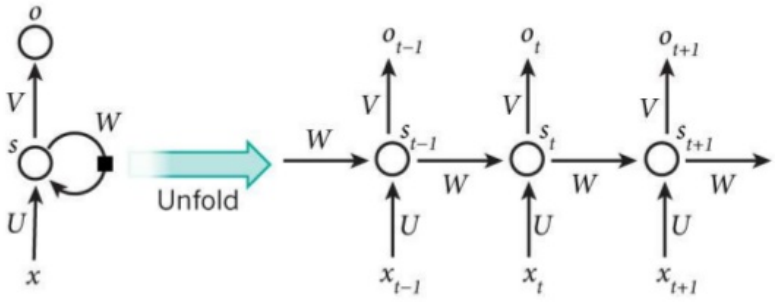

Figure 2.5: Unfolded RNN

batch is usually fixed and smaller than the total size of the dataset, and is used to perform an iteration in the weight update process (Equation 2.8). This technique is called stochastic gradient descent [4] (SGD).

Before updating the weights, the partial derivatives of the loss function are computed using the back-propagation algorithm [5]. For this we need to be able to compute the derivatives of both the loss function and the activation functions of the neurons.

After computing the derivatives, the norm of the gradient can be scaled by a factor called learning rate (expressed as λ in Equation 2.8) that can be used to speed-up or slow-down the convergence of the loss function during training. Other ways to improve this update are Adam [15] and Adadelta [18].

2.4 Recurrent networks

Recurrent neural networks (RNN) are a special type of architecture capable of modeling sequences, where the outputs of a hidden layer are fed back to the inputs of the same layer. This feedback loop introduces a state in the neurons and the output of the layers can be rewritten as:

$$\mathbf{y_t} = \sigma\left(\mathbf{W} \cdot \mathbf{x_t} + \mathbf{U_t} \cdot \mathbf{y_{t-1}}\right) \tag{2.9}$$

Where $\mathbf{x}[n]$ is an input vector at the n-th timestep and $\mathbf{U_i}$ is an additional set of tranable weights. Because of the introduction of a neuron state, RNNs can model various types of sequential data (predict the next frame of a video given the N first, translate a sequence of words to another a different language, predict the duration of each of the phonemes in a sequence, etc...).

Recurrent networks can still be trained efficiently by using back-propagation through time [17] which is specific case of back-propagation where the errors are also back-propagation back in time in the same ways that they are propagated backwards from the outputs to the inputs of the network. If we expand 2.9 we get:

$$\mathbf{y_t} = \sigma\left(\mathbf{W} \cdot \mathbf{x_t} + \mathbf{U_t} \cdot \sigma\left(\mathbf{W} \cdot \mathbf{x_{t-1}} + \mathbf{U_{t-1}} \cdot \sigma\left(\cdots \sigma\left(\cdots\right) \cdots\right)\right)\right) \tag{2.10}$$

The recursive multiplication of U_t make RNNs a special case of DNNs, where the repeated multiplications can cause the gradients to become too small or too big by the time they reach the inputs of the network when doing back-propagation. This is due to the vanishing or exploding gradient problem [2] that occurs in SGD.

To mitigate this problem in RNNs, we can use specially designed recurrent neurons such as Long Short-Term Memory as upposed to regular RNNs.

2.4.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory [11] or LSTM are a special kind of recurrent unit that deals with the aforementioned problem of the vanishing gradient. This cell works by introducing gating mechanism operated by soft switches than are trained while optimizing the network to control the flow of information coming in and out of the cell. Figure 2.6

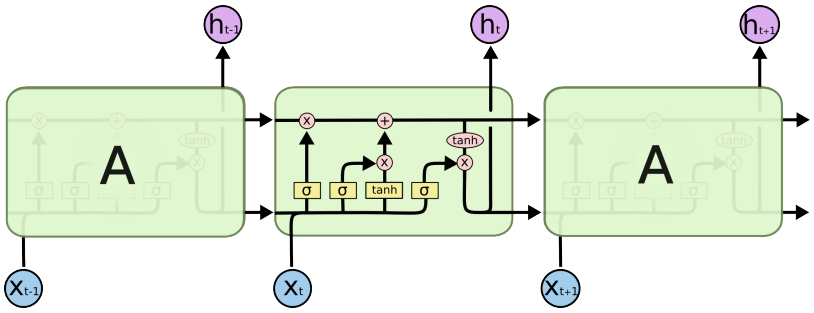

Figure 2.6: Diagram of a chain of LSTM units. The horizontal axis represents time and the vertical represents the RNN-LSTM at each timestep. X_t are the input vectors at the past, present and future timestep. The gating mechanism controls the flow of information that is input to the next state of the layer. This ffigure was taken from [**?**]

Chapter 3

Expressive Speech synthesis

3.1 Baseline development

The first stage of this project was to develop a baseline speech synthesizer. This model is used as a reference when comparing the results of the subsequent experiments explained in later sections of this chapter. The architecture is based on the work done by [8] by using the Socrates Text-to-speech framework, which is based on the Keras deep learning library. This is a RNN-LSTM based model, which, as mentioned in section 2.1.3, contains a duration model and an acoustic model which are trained independently.

Figure 3.1: Baseline model

- The duration model predicts the duration of a phoneme. This model takes a vector of linguistic features with information about the phoneme and its context and outputs a single value that is the log-compressed duration of the phoneme (this log-compression is explained in the data preparation section).

- The acoustic model predicts the excitation and spectral parameters of a frame of speech. The input of this system is the same vector of linguistic features, the normalized duration of the predicted value from the pervious model and the relative duration of the frame within the duration of the phoneme. The output of this model is also explained in the next section.

3.1.1 Data preparation

The database that has been used for this project contains approximately 20 hours or several audiobooks along with the transcriptions. The data came from the Blizzar challenge [14] and is already segmented at the utterance level, removing the need to align the whole audio stream with the raw text data. The reason we chose this database was because of its richness in expressive content. Table 3.1 shows some information about the audio files that we had.

Metric	Value
Sampling rate (Hz)	16000
Bit depth (bits)	16
Channels	mono
Length (seconds mean)	7.23
Length (seconds std)	4.52
Speakers	1

Table 3.1: Information about the Blizzard Challenge utterances.

3.1.2 Obtaining Acoustic features

As mentiones in secton 2.1.3, this RNN based speech synthesizer is a SPSS case and as such it doesn't output the waveforms directly but the excitation and spectral parameters before they are reconstructed by a vocoder. The vocoder that we used is Ahocoder [7].

Every audio file is framed by means of a sliding window with a stride of 25 milliseconds. Ahocoder then processes each frame to produce a set of excitation and spectral parameters which are:

- Mel Frequency Cepstral Coefficients (MFCC) of order $p = 39$, which corresponds to 40 coefficients.
- Pitch contour ($log(F_0)$). Unvoiced frames correspond to $F_0 = 0$ value of $-\inf$ after the logarithm. Ahocoder outputs a value of -10^8 when the frame is unvoiced.
- Maximum voiced frequency (fv), This value is 0 when a frame is unvoiced.
- Voiced/Unvoiced (UV) flag. This value indicates if a given frame of audio corresponds to a voiced or unvoiced sound and can be obtained using the fv or $log(f0)$ outputs of Ahocoder.

The data is not used as it is, at the output of the Ahocoder. These acoustic predictors are normalized before they are used to train the acoustic model for a good behavior of the back-propagation algorithm, as discussed in [8]. The outputs of the Ahocoding process have been normalized so that they are bound between a minimum and a maximum value range. This range is choosen to be $0.01, 0.99$ for better convergence properties. The normalization of the acoustic outputs is shown in equation 3.1

$$y = 0.01 + (0.99 - 0.01)\frac{y - y_{min}}{y_{max} - y_{min}} \tag{3.1}$$

This doesn't work for the $log(F_0)$ and f_v features since their high dynamic range will compress the values from the viced frames too much. [8] solves it by keeping the values from the voiced frames and interpolating the values in the unvoiced regions as shown in Equation 3.2:

$$logF_0^i = logF_0^p + (logF_0^n - logF_0^p) \cdot \frac{i - p}{n - p} \tag{3.2}$$

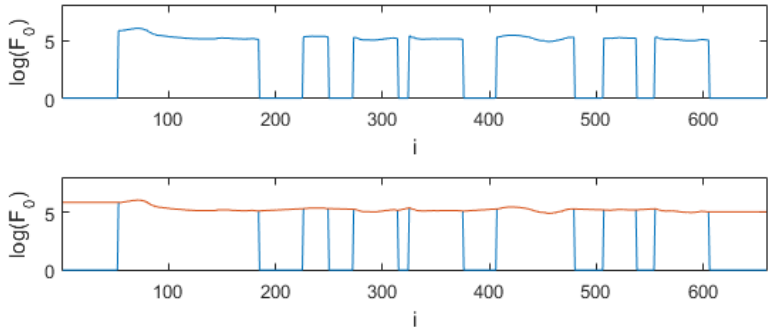

Figure 3.2: F_0 contour interpolation

Where n is the next voiced frame's frame index, F_0^n is the next voiced frame's first value, p is the previous voiced value's frame index, F_0^p is the previous voiced value and F_0^i is the i-th new interpolated value to be replaced in the original output of the Ahocoding process (Figure 3.2 has an example of this interpolation). We can then use the UV flag to recover the original format after denormalization and reconstruct the waveform.

These acoustic features are used to train the acoustic model of the system. The data is structured for training as a series of input-output vector pairs layed out in the following manner in the training data tables:

```
<linguistic features 1, duration 1> <acoustic features 1>
<linguistic features 1, duration 1> <acoustic features 2>
<linguistic features 1, duration 1> <acoustic features 3>
<linguistic features 2, duration 2> <acoustic features 4>
<linguistic features 2, duration 2> <acoustic features 5>
<linguistic features 2, duration 2> <acoustic features 6>
<linguistic features 2, duration 2> <acoustic features 7>
<linguistic features 3, duration 3> <acoustic features 8>
<linguistic features 3, duration 3> <acoustic features 9>
...
```

Figure 3.3: Training table used to train the acoustic model is stored in the server disk.

Which are the examples that are used to train the acoustic model. Some of the inputs are repeated deppending on the duration of the phonemes. The linguistic features are explained in the next section.

3.1.3 Obtaining linguistic features

Each of the audio transcriptions is transformed into a lower level phonetic representation called Label that contains a set of both real and categorical descriptors with information about each phoneme and its context within a sentence. Table **??** contains a description of the values that are included in this representation.

When a phoneme is converted into a Label, it has the following format:

p1^p2$-$p3$+$p4$=$p5~p6_p7/A:a1_a2_a3/B:b1-b2-b3~b4-b5&b6-b7#b8-b9$b10...
-b11!b12-b13;b14-b15|b16/C:c1+c2+c3/D:d1_d2/E:e1+e2~e3+e4&e5+e6#e7+e8...
/F:f1_f2/G:g1_g2/H:h1=h2~h3=h4|h4/I:i1_i2/J:j1+j2-j3

label format

Symbol	Description
p1	phoneme identity before the previous phoneme
p2	previous phoneme identity
p3	current phoneme identity
p4	next phoneme identity
p5	the phoneme after the next phoneme identity
p6	position of the current phoneme identity in the current syllable (forward)
p7	position of the current phoneme identity in the current syllable (backward)
a1	whether the previous syllable is stressed or not (0; not, 1: yes)
a2	whether the previous syllable is accented or not (0; not, 1: yes)

Table 3.2 (continued)

a3	number of phonemes in the previous syllable
b1	whether the current syllable stressed or not (0: not, 1: yes)
b2	whether the current syllable accented or not (0: not, 1: yes)
b3	the number of phonemes in the current syllable
b4	position of the current syllable in the current word (forward)
b5	position of the current syllable in the current word (backward)
b6	position of the current syllable in the current phrase(forward)
b7	position of the current syllable in the current phrase(backward)
b8	number of stressed syllables before the current syllable in the current phrase
b9	number of stressed syllables after the current syllable in the current phrase
b10	number of accented syllables before the current syllable in the current phrase
b11	number of accented syllables after the current syllable in the current phrase
b12	number of syllables from the previous stressed syllable to the current syllable
b13	number of syllables from the current syllable to the next stressed syllable
b14	number of syllables from the previous accented syllable to the current syllable
b15	number of syllables from the current syllable to the next accented syllable
b16	name of the vowel of the current syllable
c1	whether the next syllable stressed or not (0: not, 1:yes)
c2	whether the next syllable accented or not (0: not, 1:yes)
c3	the number of phonemes in the next syllable
d1	gpos (guess part-of-speech) of the previous word
d2	number of syllables in the previous word
e1	gpos (guess part-of-speech) of the current word
e2	number of syllables in the current word
e3	position of current word in the current phrase (forward)
e4	position of current word in the current phrase (backward)
e5	number of content words before the current word in the current phrase
e6	number of content words after the current word in the current phrase
e7	number of words from the previous content word to the current word
e8	number of words from the current word to the next content word

f1	gpos (guess part-of-speech) of the next word
f2	number of syllables in the previous word
g1	number of syllables in the previous phrase
g2	number of words in the previous phrase
h1	number of syllables in the current phrase

Table 3.2 (continued)

h2	number of words in the current phrase
h3	position of the current phrase in utterance (forward)
h4	position of the current phrase in utterance (backward)
h5	Phrase modality (question, exclamation, etc.)
i1	number of syllables in the next phrase
i2	number of words in the previous phrase
j1	number of syllables in this utterance
j2	number of words in this utterance
j3	number of phrases in this utterance

Table 3.2: Context-dependent label format.

A full example of this conversion is shown in **??**:

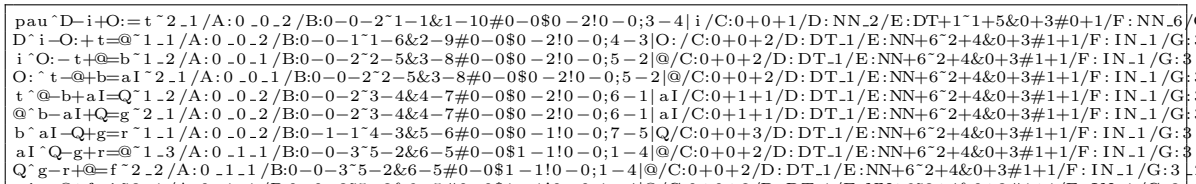

Figure 3.4: Label representation of the phrase "the autobiography of a horse."

This label conversion is carried out with the aid of the Ogmios framework [3]. This program offers a collection of speech processing modules that are combined to perform a given task. For the case of this project, it is used to obtain this label representation and to predict the duration of each of the phonemes in the sentences from the Blizzard corpus. Figure 3.5 shows the steps that it takes to obtain this information:

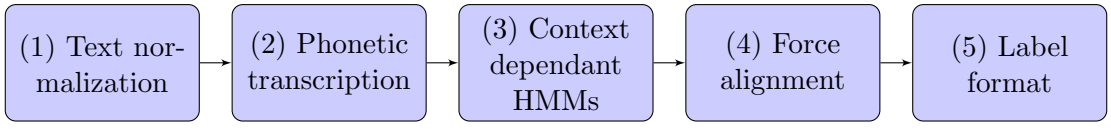

Figure 3.5: Ogmios modules used to predict phoneme duration.

1. The text normalization module converts numeric values, dates, etc to its corresponding text form.

2. The phonetic transcription module provides the pronunciation of the words.

3. The HMM module estimates a left-to-right HMM model for each of the phonemes (Figure 2.1) using the speech spectral parameters.

4. Given the phonemes of a sentence, it chains the HMM models and the most probable sequence of states is calculates using the Bitervi algorithm. With this information it can calculate the duration of each phoneme.

5. The phonetic transcriptions along with the duration predictions are converts to the label format from table **??**.

This data also has to be normalized prior to training. The label format contains both categorical features and real features. The categorical information is encoded so that all categories are orthogonal to each other by using a one-hot encoding. As a small example of a one-hot encoding, consider the three categories $\{A, B, C\}$. Its one-hot encoding would be $\{(1,0,0),(0,1,0),(0,0,1)\}$.

The real fratures are normalized so that the mean equals zero and the standard deviation equals one. This is called z-norm and the operation is shown in 3.3.

$$z = \frac{x - \mu}{\sigma} \tag{3.3}$$

As for the predicted duration of each phoneme, rather than using the raw value and as previously mentioned, it is log-compressed to avoid having a long-tailed distribution that can distort the training of the model when using the Mean Square Error (ME) minimization as discussed in [8]. Figure 3.6 contains the histogram of the raw predicted durations and the histogram after the log-compression operation.

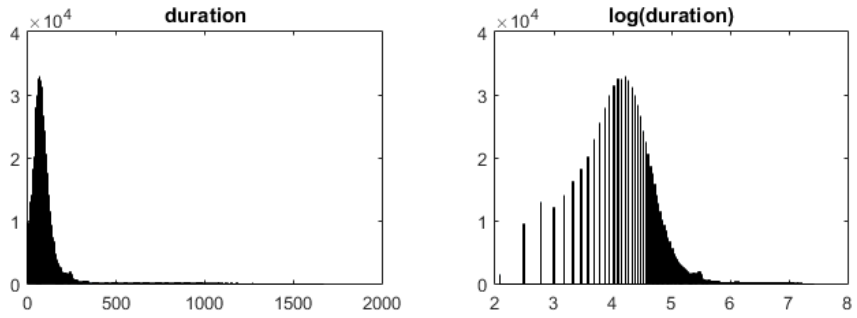

Figure 3.6: Histogram of the phoneme duration predictions by Ogmios.

Once we have ensambled the linguistic features, we discard of those who reference the past and only keep the ones tagged with an asterisk ($*$). This is because the models are based on LSTM-RNN which are known to be good at modeling long term dependencies[11].

3.2 Expressive synthesis development

The baseline system has been explained. Once we had the system up and running it was time to introduce the modifications that would allow for expressive speech to be modeled by the system. The way this is done in this project is by introducing a new set of features as inputs to the system (both the duration and acoustic model). These are known as the embedding that encapsulate the information about the way a given sentence is supposed to be uttered as. In this section we describe the process of obtaining these new features. Other than these new inputs, the developed system shares the same architecture from the baseline system.

3.2.1 Sentiment analysis & embedding extraction task

To obtain the embeddings, we define three

3.3 Experiments

Explain the experiments that were carried out and whose subsequent evaluation and results are presented in the next chapter

Chapter 4

Evaluation & Results

This chapter shows the results from the different experiments and attempts to fraw conclusions from them.

4.1 Objective evaluation

Training curves of the different stages of this project (accuracy & loss curves) as well as objective metrics from the work proposal document.

4.2 Subjective evaluation

Results from the subjective evaluation. Box plots comparing experiments.

Chapter 5

Budget

An approxmation of the budget of this project, taking into account server uptime, weekly mettings, etc..

Bibliography

[1] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[3] Antonio Bonafonte, Pablo D Agüero, Jordi Adell, Javier Pérez, and Asunción Moreno. Ogmios: The upc text-to-speech synthesis system for spoken translation. In *TC-STAR Workshop on Speech-to-Speech Translation*, pages 199–204, 2006.

[4] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[5] Yves Chauvin and David E Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology Press, 1995.

[6] Sin-Horng Chen, Shaw-Hwa Hwang, and Yih-Ru Wang. An rnn-based prosodic information synthesizer for mandarin text-to-speech. *IEEE transactions on speech and audio processing*, 6(3):226–239, 1998.

[7] E. Navas D. Erro, I. Sainz and I. Hernaez. "improved hnm-based vocoder for statistical synthesizers. In Proc. of INTER-SPEECH, editor, *Advances in Neural Information Processing Systems 29*, pages 1809–1812. Curran Associates, Inc., 2011.

[8] Santiago Pascual de la Puente. Deep learning applied to speech synthesis. Master's thesis, Escola Tecnica Superior d'Enginyeria de Telecomunicació de Barcelona, Signal Theory and Communications Dep., June 2016.

[9] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[10] Pavel Golik, Patrick Doetsch, and Hermann Ney. Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Interspeech*, pages 1756–1760, 2013.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Andrew J Hunt and Alan W Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 1, pages 373–376. IEEE, 1996.

[13] Igor Jauk, Antonio Bonafonte Cávez, Paula López Otero, and Laura Docio Fernández. Creating expressive synthetic voices by unsupervised clustering of audiobooks. In *INTERSPEECH 2015: 16th Annual Conference of the International Speech Communication Association: Dresden, Germany: September 6-10, 2015*, pages 3380–3384. International Speech Communication Association (ISCA), 2015.

[14] Simon King and Vasilis Karaiskos. The blizzard challenge 2016. 2016.

[15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[17] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[18] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[19] Heiga Zen, Takashi Nose, Junichi Yamagishi, Shinji Sako, Takashi Masuko, Alan W Black, and Keiichi Tokuda. The hmm-based speech synthesis system (hts) version 2.0. In *SSW*, pages 294–299, 2007.