Contents

Chapter 1

Introduction

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

Chapter 2

State of the art

In this section, Deep Learning and neural networks are briefly introduced. In addition to that, both the text-to-speech and sentiment analysis tasks are also introduced and discussed, including the challenges that they present as well as the classical and more modern techniques that tackle them, and how they have influenced the development of this project. By the end of this chapter, the core concepts necessary for the discussion of the work done in this publication will have been introduced to the reader.

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 give speech impaired people a voice to communicate with and it can also be used by non-sighted people by simulating more human-like interactions between them and the unanimated objects around them.

It can also have beneficial impacts from a more financial perspective, for example, by cheapening the cost of producing audiobooks or other products that require the use of real speech today.

2.1.1 Unit selection

Unit selection systems function by rendering a speech signal by concatenating waveform fragments from a large single-speaker database [9]. The outcome of this technique is highly natural-sounding speech signal since the fragments that are concatenated come from real speech waveforms. However, while this system is good at producing natural soundounding speech, it lacks the flexibility to produce new speaker sounds or vary the speaking style, as described in [10], as we are limited by the waveforms database in the first place. Each unit can also have an associate feature vector containig prosodic information such as pitch or formants.

It also has a high storage cost, since we need to have access to all of the waveform fragments to produce the speech signals.

$$asd \tag{2.1}$$

To find the best combination of units, a criterion defined in [9] is used which has a cost of concatenation of consecutive units, and a cost that compares the prosodic features of a given unit and some desired prosodic features.

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.

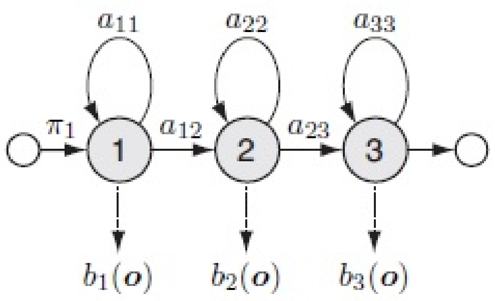

Figure 2.1: A 3-state left-to-right Hidden Markov Model.

As seen in Figure 2.1, a HMM is a finite state machine that can be described by the touple $\lambda = \langle A, B, \Pi \rangle$ 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

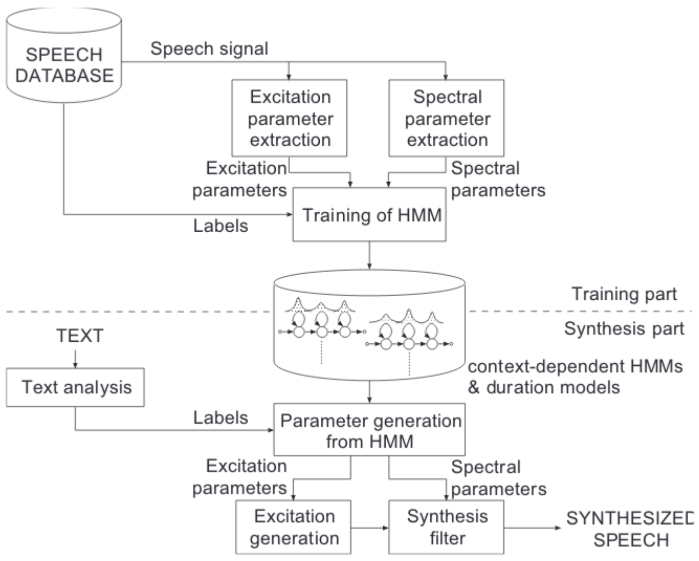

Figure 2.2: SPSS system based on HMMs.

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 apropriately as describes in [15].

In the methodology chapter of this document, the concepts from this section will be revisited and explained more in-depth.

2.1.3 Recurrent Neural Network-based Speech Synthesis

Another approach to speech synthesis that is the one used in 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 [5], this scheme consists of using neural networks to generate parametric speech in two stages: a duration model predict the duration of a phoneme, and an acoustic model to predict the spectral and excitation parameters for as many timesteps as predicted by the duration model. Figure 2.3 shows the complete scheme of this approach.

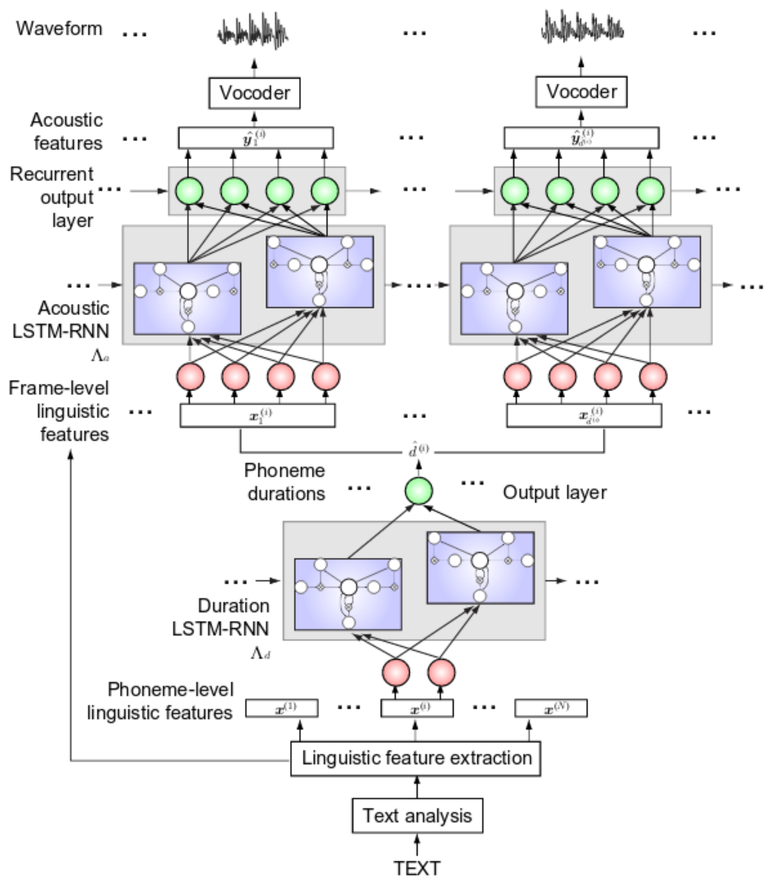

Figure 2.3: RNN-based SPSS scheme. Concepts such as LSTM that appear in the figure are covered at later section of this chapter.

2.1.4 Expressive

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

2.2 Text classification for sentiment Analysis

2.2.1 End-to-end approach

Treating text as a regular signal, using word embeddings and convolutional neural networks.

2.3 Multimodal learning

Summary of the bibliography in multimodal learning, the different combinations of modes in the bibliography and how this project is influenced by it.

2.4 Deep Learning

In recent years, deep structured learning, or more commonly called deep learning is a set of techniques that has risen in popuarity and established itself as a new area of machine learning among the scientific and research communities. [6]

[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 pupularity 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 adavantage of deep models such Imagenet [12] 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.

2.4.1 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.

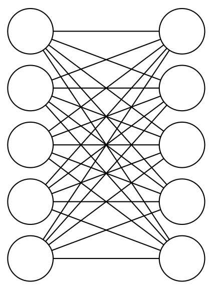

Figure 2.4: Neural network basic configuration. Each layer consists of a net of nodes chich perform a linear operation of the inputs, before being fed to an activation function at the output and fed to the next layer.

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 (2.3) 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 2.2 shows the operation 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.

$$\mathbf{y_i} = \sigma\left(\mathbf{W_i} \cdot \mathbf{x_i}\right) \tag{2.2}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

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.

2.4.2 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).

$$L(\mathbf{x};\mathbf{w}) = \sum_{\mathbf{x}_k} L_i(\mathbf{x_k}) \tag{2.4}$$

$$w_i[n+1] = w_i[n] - \lambda \frac{\partial L(\mathbf{x};\mathbf{w}[n])}{\partial w_i} \tag{2.5}$$

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 [7]). 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 2.4 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 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 2.5). This technique is called stochastic gradient descent [3] (SGD).

Before updating the weights, the partial derivatives of the loss function are computed using the back-propagation algorithm [4]. 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 2.5) 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 [11] and Adadelta [14].

2.5 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

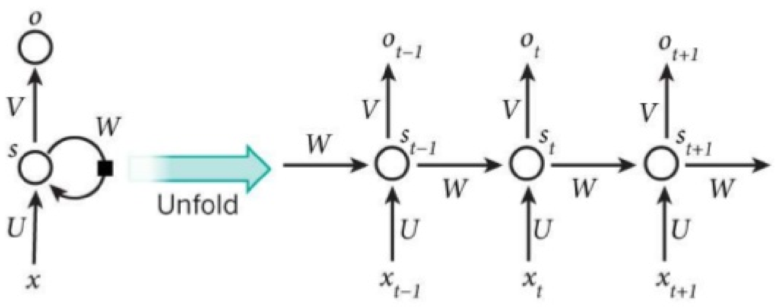

Figure 2.5: Unfolded RNN

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:

$$\mathbf{y_t} = \sigma\left(\mathbf{W} \cdot \mathbf{x_t} + \mathbf{U_t} \cdot \mathbf{y_{t-1}}\right) \tag{2.6}$$

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 phrase into a different language, etc...).

Recurrent networks can still be trained efficiently by using back-propagation through time [13] 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 2.6 we get:

$$\mathbf{y_t} = \sigma\left(\mathbf{W} \cdot \mathbf{x_t} + \mathbf{U_t} \cdot \sigma\left(\mathbf{W} \cdot \mathbf{x_{t-1}} + \mathbf{U_{t-1}} \cdot \sigma\left(\cdots \sigma\left(\cdots\right) \cdots\right)\right)\right) \tag{2.7}$$

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, as described in [2], that occurs during training with 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.

2.5.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory [8] 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

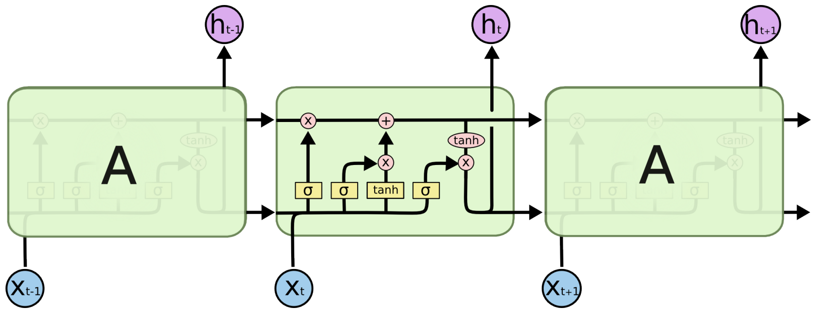

Figure 2.6: Diagram of a chain of LSTM units. The horizontal axis represents time and the vertical represents the RNN-LSTM at each timestep. X_t are the input vectors at the past, present and future timestep. The gating mechanism controls the flow of information that is input to the next state of the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

Chapter 3

Methodology

This chapter explain how the project was carried out

3.1 Sentiment analysis & embedding extraction task

Explain the two networks, the datasets used and how the data was prepared to be fed into the training process and how to obtain the experssive embeddings that will be used in the next section

3.2 Speech synthesis task

Similar to the previous section. Explain the architecture of the synthesizer, the preparation of the datasets and training process

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

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

[5] 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.

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

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

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

[9] 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.

[10] 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 Con-*

ference of the International Speech Communication Association: Dresden, Germany: September 6-10, 2015, pages 3380–3384. International Speech Communication Association (ISCA), 2015.

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

[12] 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.

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

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

[15] 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.