SpECS: Speech Enhancement for Customer Service using Recurrent Neural Network*

Adriano Henrique Rossette Leite[†‡] and Carlos Henrique Quartucci Forster[‡]
[†]Data Science Team, Itaú Unibanco, São Paulo, Brazil
[‡]IEC, Instituto Tecnológico de Aeronáutica, São José dos Campos, Brazil
Email: [†]contact@adrianohrl.tech, [‡]forster@ita.br

***Abstract*— This work conducts a study on the characteristics of present noise signal in telephone calls in customer service. In order to minimize this problem, an speech enhancement model, named as SpECS, for phone support is proposed. SpECS stands for Speech Enhancement for Customer Service and uses a encoder-decoder long short-term memory recurrent network for eliminating and reducing ordinary noise found in a phone call between a company's client and attendant. The dataset generation is detailed. It is verified that, even though the possibilities to apply data augmentation in such application is wide, the problem turns intractable.**

***Index Terms*—Costumer Service, Speech Enhancement, Long Short-Term Memory Recurrent Neural Networks, Deep Learning.**

I. Introduction

As technology advances, it makes feasible to companies storing data related to their relationship with costumers for data mining [1][2][3]. There are different types of costumer service channels, of which the main ones are: self-service knowledge base, social media, live chat, email, and phone support [4]. The last one is probably the oldest form of customer service available today. It is a voice transmission system [5] that links a costumer to an attendant. Its schema consists of a pair of microphone and speaker for the customer, a bidirectional channel, and another pair of microphone and speaker for the attendant. All these components may be too noisy. As a consequence, the customer service is impaired, leading dissatisfaction to the company's clients.

As companies aim to maximize the satisfaction of their clients [6], the existing noise in the costumer service via phone support should be minimized. The noise found in a voice transmission system comes from: (1) *noisy client and attendant's microphone* (2) *noisy channel*, and (3) *noisy client and attendant's environments*. At first, the quality of the signal generated by client and attendant's microphone are influenced by its sensor precision. Secondly, the noise coming from the channel is due to energy dissipation, low quality connectors, and physical interference. These sources cause one of the following types of audio noise:

- *white noise*: is a signal made of uncorrelated samples, such as the numbers produced by an uniform or Gaussian random generator, all frequencies has the same proportion in the spectrum;
- *pink noise*: is a random signal, filtered to have equal energy per octave. In order to keep the energy constant over octaves, the spectral density needs to decrease as the frequency (f) increases. This decrease corresponds to 3 dB per octave on the magnitude spectrum;
- *brown noise*: is a random signal that has been filtered in order to generate a lot of energy at low frequencies. Its power density is inversely proportional to the quadratic frequency (f^2) and decreases by 6 dB per octave. Brown noise produces a much warmer tone than white noise (0 dB/oct) or pink noise (-3 dB/oct).
- and *gray noise*: is a random noise that feels perceptually flat.

Lastly but not least, the client and attendant's environments may be too noisy. The common environments that clients are inserted are related to traffic, crowded places, children, object crashing, and others. On the other side of the channel, the attendant's environment is usually a call center with many other attendants talking to other clients. This last type of source is more complex than the previously cited ones.

The problem of enhancing speech that has been corrupted by noise is composed by two challenges: the accurate estimation of the noise that corrupts any particular segment of speech, and the effective attenuation of this noise without diminishing the speech in the signal [7]. This denoising process could be applied in a costumer service phone support to avoid repetitions requested by the attendant to the client due to the bad comprehension. In this case, the client's original signal would be processed before sampling it to the attendant speaker. Another possible application is, instead of storing the original speech in the company's data lake, it would be better to store the enhanced one to improve its quality score. For that, both client and attendant's speeches would be processed before storing them in the company's data lake.

In 2015, Osako *et. al.* proposed a recurrent gated structure neural network design that integrated the two challenges of speech enhancement in a single process [7]. The proposed algorithm can recover much of the degradation caused by the noise.

A model named as SpECS for speech enhancement in a costumer service through phone support is proposed in this work. SpECS stands for Speech Enhancement for Costumer

*Any opinions, findings, and conclusions expressed in this manuscript are those of the authors and do not necessarily reflect the views, official policy or position of the Itaú-Unibanco.

Service and it uses a Long Short-Term Memory Recurrent Neural Network (LSTM RNN) to estimate and eliminate the characteristic noisy signal present in the costumer and attendant's speech.

The rest of this work is organized according to the following description. The Section **??** presents other contributions related to denoising speech signals. Then, the Section II conducts a review on Encoder-Decoder Long Short-Term Memory Recurrent Networks. The Section III states the main considerations adopted during dataset generation and model design. In the Section IV, this work results are shown and analyzed. Finally, the conclusions are presented in Section V.

II. Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNN) are processing structures that are capable to model dynamic behaviours [8][9]. This is possible by replicating the output of a block of neural nodes to its own input. Thus, the information of past inputs can be propagated to process future inputs. This structure allowed sequence problems to be handled in Deep Learning. However, the training process of a RNN has two issues: the exploding and vanishing gradient issue [10]. The *exploding gradients* problem refers to the large increase in the norm of the gradient during training. While the *vanishing gradients* problem refers to the attenuation of long term components.

In order to minimize these problems, the Long Short-Term Memory (LSTM) was proposed [11]. The following subsection reviews the LSTM and the Encoder-Decoder LSTM structures.

A. Long Short-Term Memory RNN (LSTM)

The Figure 1 shows the original configuration [11] of the Long Short-Term Memory Recurrent Neural Network (LSTM RNN) memory cell. Its upper horizontal line is the main idea behind LSTM RNN [12]. It illustrates the two operations that are processed during each iteration to update the memory state. Based on the input data, the past information that became irrelevant is forgot from memory and, then, new relevant information is stored in memory. Finally, the updated memory state is used to make new predictions. To sum up, these two operations keep only relevant information in memory as new information arrives. Each LSTM component will be detailed in the next paragraphs of this subsection.

The LSTM structure has a component called *forgetting gate*. This part of the memory cell is responsible for generating a signal vector based on the previous prediction (y_{k-1}) and the incoming information (x_k). Each element of the forgetting signal (f_k) is in range of $[0;1]$ so that it will keep relevant information in memory ($\tilde{m}_k$) and attenuate information that become irrelevant. The forgetting signal is defined in Equation 1. The values of the weight matrix W_f and the bias vector b_f are learned during the model training phase.

$$f_k = \sigma\left(W_f \cdot [y_{k-1}, x_k] + b_f\right) \tag{1}$$

As Equation 2 states, the element-wise multiplication between the forgetting signal (f_k) and the previous memory state

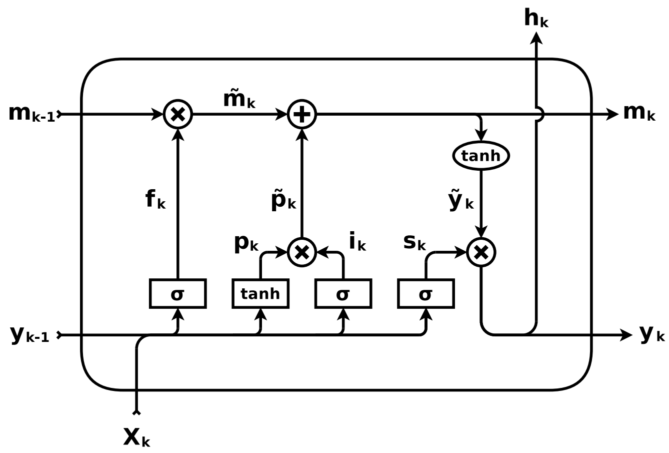


Fig. 1: The original configuration of the Long Short-Term Memory Recurrent Neural Network (LSTM) memory cell, proposed in [11]. In this LSTM diagram: (1) lines represent data transportation, (2) lines directions correspond to the one in which the data flow, (3) lines merging denote data concatenation, (4) a line forking denotes data replication, (5) ellipses represent element-wise operations, and (6) rectangles represent neural network layers.

(m_{k-1}) results in a filtered signal ($\tilde{m}_k$) that contains only relevant information from the past.

$$\tilde{m}_k = f_k * m_{k-1} \tag{2}$$

All possible prediction (p_k) are estimated by a neural network layer given the previous prediction (y_{k-1}) and the incoming information (x_k) as its input. The default activation function in this neural network layer is a hyperbolic tangent ($\tanh$), but other activation function may also be used. The Equation 3 defines the possible prediction signal. The values of the weight matrix W_p and the bias vector b_p are learned during the model training phase.

$$p_k = \tanh\left(W_p \cdot [y_{k-1}, x_k] + b_p\right) \tag{3}$$

Another important part of the LSTM memory cell is the *ignoring gate*. It is similar to the forgetting gate because it also generates a signal vector based on the previous prediction (y_{k-1}) and the new information (x_k). In addition, each element of the ignoring signal (i_k) is in range of $[0;1]$, too. Nevertheless, its role is quite different. It keeps relevant prediction possibilities ($\tilde{p}_k$) and bypasses the irrelevant ones. The ignoring signal is defined in Equation 4. The values of its weight matrix (W_i) and bias vector (b_i) are learned during the model training phase.

$$i_k = \sigma\left(W_i \cdot [y_{k-1}, x_k] + b_i\right) \tag{4}$$

The element-wise multiplication between the ignoring (i_k) and the possible prediction (p_k) signals results in a filtered signal ($\tilde{p}_k$) that represents only relevant prediction possibilities. This is stated at Equation 5.

$$\tilde{p}_k = i_k * p_k \quad (5)$$

The LSTM memory state (m_k) is updated when the filtered possibilities ($\tilde{p}_k$) is added to the relevant information from past ($\tilde{m}_k$), just as the Equation 6 shows.

$$m_k = \tilde{m}_k + \tilde{p}_k \quad (6)$$

After this update, a normalized relevant information signal ($\tilde{y}_k$) is generated by applying an element-wise hyperbolic tangent ($\tanh$) operation to the current memory state signal (m_k). This is represented by Equation 7.

$$\tilde{y}_k = \tanh(m_k) \quad (7)$$

The *selecting gate* is another part of the LSTM memory cell that generates a signal vector based on the previous prediction (y_{k-1}) and the new information (x_k). Each element of the selection signal (s_k) is in range of $[0; 1]$ so that it filters the normalized memory state to let only the predictions pass to the output. The selecting signal is defined in Equation 8. The values of the weight matrix W_s and the bias vector b_s are learned during the model training phase.

$$s_k = \sigma(W_s \cdot [y_{k-1}, x_k] + b_s) \quad (8)$$

Finally, the LSTM resulting prediction (y_k) is outputted by an element-wise multiplication between the selecting (s_k) and the normalized memory state ($\tilde{y}_k$) signals, as the Equation 9 shows.

$$y_k = s_k * \tilde{y}_k \quad (9)$$

B. Encoder-Decoder LSTM

The LSTM network can be organized into an architecture called the Encoder-Decoder LSTM. This neural network structure deals with sequence-to-sequence data problems. In other words, an Encoder-Decoder LSTM allows the model to be used to both support variable length input sequences and to predict or output variable length output sequences [12].

This architecture comprises two models: one for reading the input sequence and encoding it into a fixed-length vector, and a second for decoding the fixed-length vector and outputting the predicted sequence [13]. Encoder-Decoder LSTM is the basis for many advances in complex sequence prediction problems such as text translation and speech recognition [14].

III. SpECS

The model formulation considers that its input and output data is a noisy speech and an enhanced version of the equivalent speech, respectively. This is classified as a sequence-to-sequence problem, in which the input and output length may vary and the temporal ordering of the observations matters.

This section presents the generated dataset used during the train and test phases for model generation and validation. In addition, it also describes the model hyper-parameters.

A. Datasets

The model input data was generated from the output data. That is, the noisy speech was obtained by adding noise to a clean audio corpora.

For this, the Brazilian Federal Constitution Speech Corpora[1] was to generate the model output data. This corpora is composed of a man announcing the Brazilian Federal Constitution in Portuguese during approximately 9 hours of speech inside a controlled recording environment. The audio was resampled to 22.050 Hz with 16 bits. Finally, it was segmented into smaller files, in which each of them lasts approximately 30 seconds.

The sampling rate of the chosen corpora is not equivalent to the one used in telephony. In this context, the usable voice frequency band ranges from approximately 300 Hz to 3400 Hz [15]. Per the NyquistShannon sampling theorem [16], the sampling frequency (8 kHz) must be at least twice the highest component of the voice frequency via appropriate filtering prior to sampling at discrete times (4 kHz) for effective reconstruction of the voice signal. For this reason, the whole corpora was resampled to 8 kHz. By doing this proceeding, the speech became less brighter since high frequencies cannot be sampled anymore. As a consequence, the size of the manipulated data was also reduced. Besides the quality of the resampled audio approached the ordinary quality of a telephone call speech. This resampled corpora was used as the desired output of the model because now its recording quality is feasible in a phone support application.

Therefore, the input signal is equals to the output added to a characteristic noise. As reviewed in ..., the characteristic noises in customer service for telephone calls ...

The generated dataset was splitted into two parts: the training and the test datasets. On one hand, the training dataset corresponds to 70% of the whole dataset and it is used to create the model. On the other hand, the test dataset is used to validate the model's capability of generalization and corresponds to the remaining 30% of whole dataset.

B. Model

Since this is a sequence-to-sequence problem, an encoder-decoder LSTM RNN was used to represent the speech enhancement model. The chosen neural network configuration is capable of learning patterns given a sequential data as input to generate a sequential data as output. The learned pattern is transformed into an encoded representation, which is decoded to reproduce an output sequence.

As audio data oscillates between positive and negative, the activation function ReLU was not used in the neural network layers shown in Figure 1. The hyperbolic tangent was chosen an the neural network layers' function activation in the SpECS model.

IV. Results

This section presents the main result obtained in this work. At first, it states the setup configuration used in the used ma-

[1]Source: https://drive.google.com/uc?export=download&confirm=waHs&id=1Nf849u-27CYRzJqedLaI-FaZfMRO7FT_. Date: December 5th, 2018.

chine. Then, the results obtained during the dataset generation are analyzed. Finally, the model training phase is analyzed.

A. Setup

The extracted results of the SePCS project was obtained by an Ubuntu 16.04 virtual machine running on an 8 vCPUs, 52 GB of RAM memory system at Google Cloud Platform. At first, it was configured to run the following software packages:

- Python 3.5.1;
- Anaconda 4.0.0;
- NumPy 1.15.4;
- Pandas 0.18.0;
- LibROSA 0.6.2;
- Scikit Learn 0.17.1;
- Tensorflow 1.12.0;
- and Keras 2.2.4.

Alongside this, the resampled output data (described in Subsection III-A) was uploaded to Google Storage (1, 255 items, totalling 1.0 GB). After completing both the machine setup and the data upload, the SpECS project, hosted on the GitHub[2], was cloned in the machine. Finally, the uploaded data was ingested to a folder called *datasets/output* located on the project's root path in the virtual machine.

B. Datasets

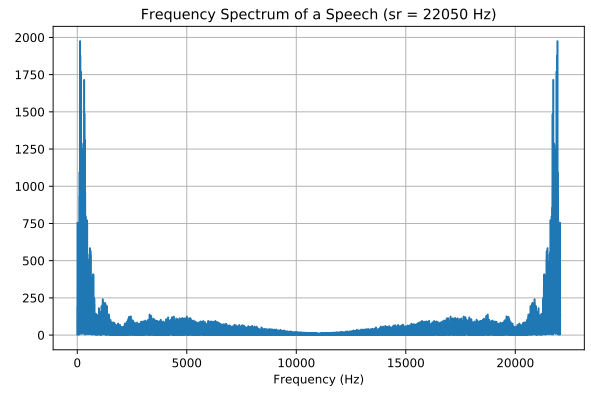
After this setup, the input data was generated as described in III-A. A random uniform distribution generator was used to generate white noisy speeches as the model input data. Given a positive value (a), the random uniform generator will generate a white noise in range of $[-a; a]$. These values were also generated randomly in an uniform distribution. The range of this generator was chosen ($[0.001; 0.05]$) in such a way it would be perceptually existent for human audition. The generated noisy speeches were stored in the *datasets/input* folder, which is located in the project's root path.

The Figure 2 shows the frequency spectrum of a given speech in the dataset during all the three phases it passed through. At first, the Figure 2a shows its frequency spectrum when in the original state, that is, sampling rate equals to 22050 Hz of a clean speech. Secondly, the Figure 2b shows its frequency spectrum when it was resampled, that is, sampling rate equals to 8000 Hz of a clean speech. Lastly, the Figure 2c shows its frequency spectrum when in the original state, that is, sampling rate equals to 8000 Hz of a noisy speech. Note in this last case that all frequencies in the spectrum had its gain equally increased. This evidences the presence of white noise on the speech. The final dataset comprises only white noisy speeches.

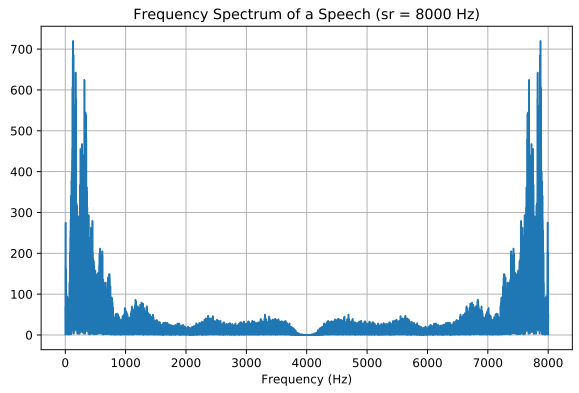
C. Model Training

SpECS is composed by the following configuration: an input layer, an episodic LSTM layer, a sequential LSTM layer, and an output layer. The episodic LSTM does not returns a sequence on its output. Its sequence data comes

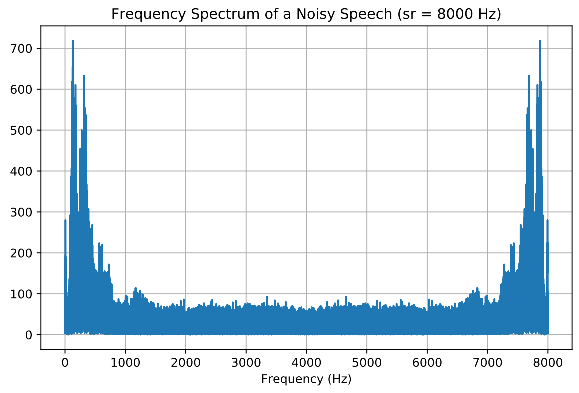
[2]https://github.com/adrianohrl/SpECS



(a) Clean speech with sampling rate equals to 22050 Hz.



(b) Clean speech with sampling rate equals to 8000 Hz.



(c) Noisy speech with sampling rate equals to 8000 Hz.

Fig. 2: The frequency spectrum of an instance of speech in each step of the dataset generation.

from its input. Oppositely, the sequential LSTM does return a sequence on its output. Nevertheless, its input is fixed and each generated output is propagated sequentially. The dropouts avoids overfitting.

Since SpECS has a complex configuration, it has many hyper-parameters. The SpECS' main hyper-parameters are described as follows.

- *input_size*: specifies the maximum size of the input audio, the chosen value was 32000 which corresponds to 4 seconds of audio at 8 kHz;
- *text_size*: specifies the proportional size of the validation data set, the chosen value was 0.3;
- *seed*: specifies the random state during the dataset split, the chosen value was 42;
- *encoder_n_cells*: specifies the number of nodes in the neural network layers of the encoder LSTM, the chosen value was 128;
- *encoder_activation*: specifies the activation function in the neural network layers of the encoder LSTM, the chosen value was *tanh*;
- *encoder_dropout*: specifies the encoder LSTM's dropout rate, the chosen value was 0.125;
- *decoder_n_cells*: specifies the number of nodes in the neural network layers of the decoder LSTM, the chosen value was 128;
- *decoder_activation*: specifies the activation function in the neural network layers of the decoder LSTM, the chosen value was *tanh*;
- *decoder_dropout*: specifies the decoder LSTM's dropout rate, the chosen value was 0.125;
- *optimizer*: specifies the optimizer to be used during the model train, the chosen optimizer was *Adam*;
- *learning_rate*: specifies the optimizer learning rate, the chosen value was 0.001;
- *loss*: specifies the metric for loss calculation, the chosen metric was *MSE*;
- *epochs*: specifies the number of epochs during the model training phase, the chosen value was 20;
- *batch_size*: specifies the number of samples to be used in each iteration during the model training phase, the chosen value was 10;
- *patience*: specifies the number of epochs to early stopping, the chosen value was 0;
- *min_delta*: specifies the minimum difference between two consecutive metrics to trigger the early stopping patience, the chosen value was 0;
- and others.

Considering the above configuration, each epoch ran approximately 40 minutes. It is too slow, even the specified number of epochs is not sufficient to obtain good results.

Many simplified configurations were tested, but it was verified that they were too simple, causing underfitting. Nevertheless, when tuning the above hyper-parameters generously, the machine resources could not deal with the amount of data and parameters to train.

V. Conclusion

The possibilities to apply data augmentation technique is wide. However, the amount of data in audio application grows rapidly. Even though, the audio files were resampled and resourceful machinery was chosen, the model could not be trained in a timely manner.

As future works, it is suggested to use machines with higher computational power (resourced with GPUs and TCUs) to train SpECS.

References

[1] E. Breck and C. Cardie, "Opinion mining and sentiment analysis," in *The Oxford Handbook of Computational Linguistics 2nd edition*, 2017.

[2] A. Schmidt and M. Wiegand, "A survey on hate speech detection using natural language processing," in *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, 2017, pp. 1–10.

[3] J. Padmanabhan and M. J. Johnson Premkumar, "Machine learning in automatic speech recognition: A survey," *IETE Technical Review*, vol. 32, no. 4, pp. 240–251, 2015.

[4] D. Leffel. (2018) 5 types of customer service. [Online]. Available: https://blog.crewapp.com/c/resources/types-customer-service/

[5] B. Goode, "Voice over internet protocol (voip)," *Proceedings of the IEEE*, vol. 90, no. 9, pp. 1495–1517, 2002.

[6] N. Hill and J. Alexander, *The handbook of customer satisfaction and loyalty measurement*. Routledge, 2017.

[7] K. Osako, R. Singh, and B. Raj, "Complex recurrent neural networks for denoising speech signals," in *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2015 IEEE Workshop on*. IEEE, 2015, pp. 1–5.

[8] A. C. Tsoi and A. Back, "Discrete time recurrent neural network architectures: A unifying review," *Neurocomputing*, vol. 15, no. 3-4, pp. 183–223, 1997.

[9] O. Nerrand, P. Roussel-Ragot, D. Urbani, L. Personnaz, and G. Dreyfus, "Training recurrent neural networks: Why and how? an illustration in dynamical process modeling," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 178–184, 1994.

[10] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[11] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," in *Advances in neural information processing systems*, 1997, pp. 473–479.

[12] C. Olah. (2015) Understanding lstm networks. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[13] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
[15] I. R. Titze and D. W. Martin, “Principles of voice production,” 1998.
[16] A. J. Jerri, “The shannon sampling theoremits various extensions and applications: A tutorial review,” *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1565–1596, 1977.