Detection of Fillers in Conversational Speech

MSc Natural Language Processing M1 Supervised Project Realization Report

by

Eduardo Calò, Thibo Rosemplatt

Supervisor:

Imran Sheikh

Reviewer:

Christophe Cerisara

Academic year 2019-2020

Contents

Acknowledgements

We would like to wholeheartedly thank Dr. Imran S. for guiding and supporting us through the whole route towards this final report. His willingness to share his knowledge, his commitment and interest to this work, and his feedback have always been essential.

Chapter 1

Introduction

This report presents our efforts and experiments for creating a classifier capable of automatically identifying and tagging events in conversational speech.

This study can be set in the more general framework of improving conversational Automatic Speech Recognition (ASR) systems. Collecting speech datasets for training ASR systems can incur high costs and difficulties, since large amount of time and labor force are needed for recording and annotating. This problem is now being partially mitigated by read speech corpora, thanks to crowdsourced projects such as Common Voice Project by Mozilla [19], LibriSpeech [20] and others. Obtaining suitable conversational speech corpora is still difficult, and creating ad-hoc datasets is an even more time-consuming and labor intensive task.

Spontaneous conversational speech is known to be characterized by the presence of disfluencies. This widespread phenomenon, common in all languages, plays an important role in structuring speech. ASR systems trained on read speech are not robust enough to handle these events, which are not present in training corpora. The best way to handle disfluency events in speech would be training a conversational ASR system exclusively using spontaneous conversational speech data, but given the aforementioned issues, alternative solutions have been proposed. One of them is automatically tagging events in speech, given a small amount of annotated conversational speech data, which is the final aim of this project.

A number of different approaches have been proposed so far to address this problem. Prosodic features have been explored [10, 12, 3] and information provided by Language Model have been also exploited [24, 25, 22, 14]. However, the most successful approach is the one proposed by Brueckner et al. [6], in which application of deep Bi-directional Long-Short-Term Memory (BLSTM) Recurrent Neural Networks (RNN) for detection and classification of events in speech is investigated.

Our attempts are along the lines of the latter.

The rest of this report is structured as follows. In Section 2 our experiments on sequence classification, made as preparation for the more complex sequence tagging problem, will be presented. In Section 3 our efforts for tagging events in speech will be presented. We will describe the methodology we followed, the pre-processing that was necessary to be done on the corpus and our RNN approach. Finally we will conclude with what we learned in this project.

Chapter 2

Towards a sequence classifier on spoken utterances

2.1 Motivation

Sequence classification task aims at predicting the category of a data sequence, such as a text sentence or an audio-video signal [15]. For example, given a sentence with a sequence of words, a sequence classifier can classify this sentence into 1-of-N topics. Similarly, a sequence classifier on spoken audio utterances can be used to classify the gender of the speaker. On the other hand, sequence tagging task aims at assigning a categorical label to each of the members of the input sequence [15]. For example, given a sentence with a sequence of words, a sequence tagger can assign a part-of-speech tag to each word in the input sentence. Similarly, a sequence tagger on spoken conversations can tag different speakers in the same audio recording [2].

The final aim of our project is to develop a sequence tagger for tagging filler like events in spoken utterances using a modern RNN approach. However, before engaging in this relatively difficult task we attempt sequence classification on spoken utterances, as a simplification of our final problem.

We begin with the task of gender classification on spoken utterances to get introduced to RNN based sequence classification. This task consists in extracting relevant features from the spoken utterances and assigning a category, masculine or feminine, to the whole input utterance.

2.2 Gender classification on spoken utterances

We attempt gender classification on spoken utterances using a recurrent neural network (RNN). RNN is a type of artificial neural network designed to handle sequential data [11]. RNNs can be said to have a 'partial memory', as they make use of history corresponding to past inputs in the sequence, in order to process the current input in the sequence. This characteristic makes them more suitable for processing sequences of arbitrary lengths, such as text and audio.

In order to understand an RNN model, let us consider an input sequence $X = \{..., x_{t-2}, x_{t-1}, x_t, x_{t+1}, ...\}$. Here t denotes index within the input sequence, for instance time steps in an audio-video signal or word in text. Typically, the input sequence is passed left to right into the RNN and for each time step t a new hidden state h_t is computed based on the previous hidden state h_{t-1} and the current input x_t. The computation of hidden states h_t in a typical Elman RNN [9] relies on the RNN parameters W^X and W^H, which are linear transformation weight matrices and used as follows:

$$h_t = \sigma(W^H h_{t-1} + W^X x_t + b^H) \tag{1}$$

where, σ denotes the sigmoid non-linearity and b^H denotes a bias vector included in the computation. This operation can be illustrated using the block diagram in Figure 1.

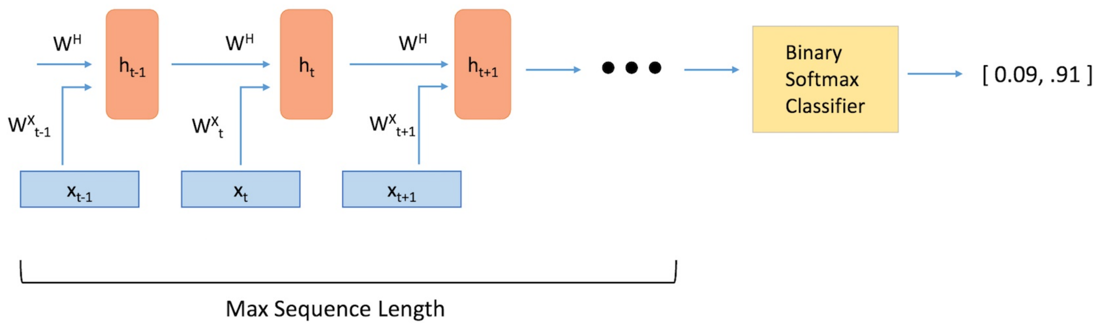

Figure 1: RNN architecture for sequence classification.

2.2.1 Model description

Figure 1 shows the architecture of our model for gender classification on spoken utterances. Audio signal corresponding to a spoken utterance is converted to a sequence of vectors of Mel Frequency Cepstral Coefficients (MFCC) [7]. These MFCC feature vectors are extracted on short duration overlapping time frames, across the temporal audio signal, and form the sequential input $\{x_0, ..., x_{t-1}, x_t, x_{t+1}, ..., x_T\}$ to the RNN model. A hidden state h_T is obtained at the end of the input sequence, once all the T frames of the input sequence have been processed. The last hidden state h_T is fed into an output transformation layer, with weight and bias parameters W^Y and b^Y:

$$y_T = W^Y h_T + b^Y \tag{2}$$

Output y_T is fed to a softmax function, ultimately resulting into probabilities p^c for each of the two output classes c, as:

$$\begin{align} p^c &= \text{softmax}(y_T)_c \tag{3} \\ &= \frac{\exp(y_T^c)}{\sum_k \exp(y_T^k)} \tag{4} \end{align}$$

Note that different spoken utterances are expected to have different time duration and hence different length T of the input sequence of MFCC feature vectors. In order to facilitate faster and efficient mini-batch training [4] of our RNN model, it is required to fix the length of sequences in each mini-batch. To address this issue we fixed the length of a mini-match to that corresponding to the longest utterance in our corpus. Input sequences shorter than this length are padded with zeros, following the standard zero padding procedure.

2.3 Experiments and results

2.3.1 Corpus

We conducted our experiment using LibriSpeech Corpus [20]. We chose the *test-clean* dataset which includes 5.4 hours of data distributed on 2620 audio files with a sampling rate of 16kHz, comprising spoken utterances of 40 speakers (20 men and 20 women). The mean, minimal, and maximal duration of the audios are respectively 7.42, 1.29, 34.96 seconds.

2.3.2 Configuration

We used MFCCs extracted with 25 ms frame size and 12.5 ms shift. From each frame we extracted a vector containing 13 MFCC coefficients. The number of MFCC vectors will be equal to the number of frames in the audio signal. Our RNN follows the architecture shown in Figure 1.

2.3.3 Results

Among the 2620 audio files available, 80% (2096) were used for training and 20% (524) for testing. We trained the model through 50 epochs, using mini-batches of 32 and a learning rate of 0.05. We used Cross-entropy loss function [8] and Adam optimizer [13]. The model achieved a train accuracy of 59.68% and a test accuracy of 58.75%. The confusion matrices relative to these accuracy values are shown in Figure 2 and Figure 3.

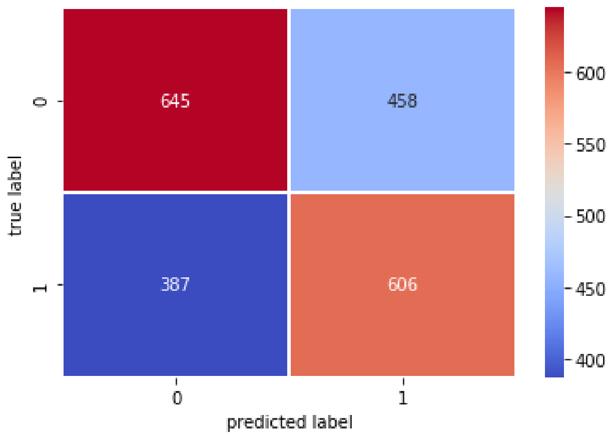

Figure 2: Confusion matrix for the train set.

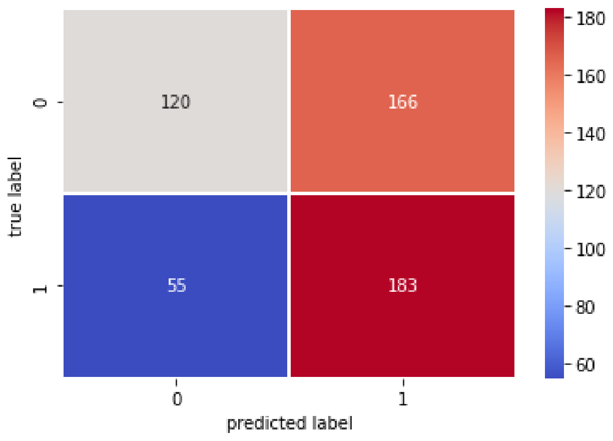

Figure 3: Confusion matrix for the test set.

These results are not satisfying for a classification task, but they are still encouraging to the extent that we succeeded in implementing a sequence classifier built over an artificial neural network, applied to spoken utterances.

This experiment taught us a lot, and was rewarding under different aspects. First, we learnt to use some of the tools mentioned in our bibliography report, like *LibROSA* [18], and the Python libraries *PyTorch* [21] and *Scikit-learn* [1]. Second, we saw for the first time what MFCCs look like, and we gained a better understanding of this feature. And finally, we got introduced to a neural network architecture. All these elements will be further helpful in our attempt on tagging fillers in conversational speech.

Chapter 3

Tagging events in speech utterance

Tagging events in conversational speech is the final aim of our project. There are several different events occurring in spoken utterances that can be tagged, such as laughter, cough, breathing, hesitations, noise, etc. In particular, in our project, we focus on three different events: speech, fillers and silence. Similar to Section 2, an RNN will be used to address this task.

3.1 Model description

The RNN model for sequence tagging is similar to the one for sequence classification, discussed in Section 2.2. The major difference between the two is the way outputs are obtained. In RNN for sequence classification, a single output is obtained after all the frames in an input sequence have been processed, and a single label is finally assigned to the whole input. Conversely, in RNN models for sequence tagging, an output label is obtained for each frame.

To better understand RNN models for sequence tagging, let us consider an input sequence $X = \{..., x_{t-1}, x_t, x_{t+1}, ...\}$, where t indicates the index of a frame in the input sequence. This input sequence is passed into the model. For each time step t, a new hidden state h_t is calculated based on the information carried by the previous hidden state h_{t-1} and the current input x_t. In addition to hidden states, an output sequence $O = \{..., o_{t-1}, o_t, o_{t+1}, ...\}$, with outputs corresponding to each time frame, is obtained for sequence tagging. This operation is described in Figure 4. In our task, input sequence X will be composed of MFCC feature vectors extracted over short overlapping frames, while output sequence O will contain the categories (speech, silence or hesitation) predicted by the model for each frame of the input sequence.

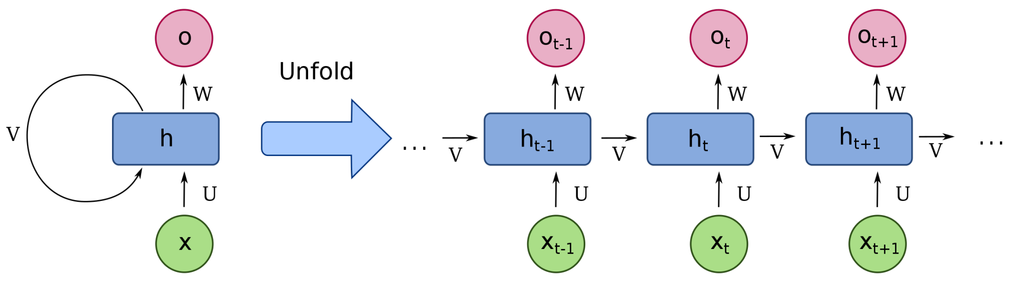

Figure 4: RNN architecture for sequence tagging.

3.2 Experiment setup

3.2.1 Corpus description

We conducted experiments using the *CallHome English* corpus of telephone conversations, made available by the the TalkBank project [26]. It consists of 120 unscripted telephone conversations between native speakers of English. The conversations last up to 30 minutes, with transcripts available for contiguous 5 or 10 minute duration segments. The transcripts are timestamped by speaker turns (as shown in Figure 5). Timestamps are of form *begin_end*, where *begin* and *end* are in milliseconds.

```
*B: &=lipsmack it's on the corner of Columbia and Cole. 206830_209640

*A: uhhuh. 210130_210500

*B: and it's a one bedroom and . 210850_212500

*B: so I had to put down like a deposit and &um . 213050_215590

*B: you know and pay the rent for it so I could hold it. 216580_218860

*A: mhm. 219150_219560

*B: but that's okay. 219880_220680
```

Figure 5: Sample of a transcript of a conversation from *CallHome*.

The corpus is accompanied with a set of transcription rules, that help us understand and process the data. In particular, the finite set of hesitation sounds used in the transcripts is provided, which will allow us to feed our model with well-annotated data.

3.2.2 Preparation for model training

As mentioned in Section 3.2.1, transcripts provide timestamps for each speaker-turn, which often comprise several words (see Figure 5). However, the model requires frame level label alignments. To this end we use a forced alignment technique which is described in Section 3.2.2.2. Before doing so we need to clean the transcripts as discussed in Section 3.2.2.1.

3.2.2.1 Corpus cleaning

The first step was to clean the transcript files in order to get a list of tuples where the first element is the cleaned transcript of one speaker's turn, and the second is the time stamp of this utterance. The original transcripts need to be cleaned to remove unwanted meta-information. This has been performed using regular expressions, which handle different cases.

After obtaining tuples of cleaned transcripts and corresponding timestamps, the original audio files were segmented using the *AudioSegment* module from the *Pydub* [23] library. We decided to exclude all utterances which do not contain a filler, in order to simplify the tagging task. However, it should be noted that this does not resemble a real scenario and must be studied separately.

3.2.2.2 Forced alignment

The process of automatically giving time stamps to every word in a speech file, given its transcription, is called forced alignment. We use force alignment to obtain frame level labels on our speech data. *Montreal Forced Aligner* [17, 16] tool was used for this step. This tool can only be used on 16kHz audio files, so we had to artificially upscale all of our audio files from 8kHz to 16kHz. This approach is not recommended and susceptible to alignment errors. Ideally one should use an alignment acoustic model trained on 8 kHz speech. Moreover, we had to convert the original stereo audio files, containing one speaker on each track, into mono audio files.

Given an audio file and its transcription, the aligner output files in the *TextGrid* format [5], which contains word level and phone level time alignments. We picked a sample of 20 aligned utterances of different length to manually assess the quality of the alignment, and we found the alignments to be reliable. Unfortunately some of the files could not be aligned. Typically, this happened when two speaker's voices were overlapping. We had to discard these utterances from the experiment.

This filtering and alignment process left us with 1.78 hour of aligned data, distributed over 2884 utterances. The mean, min, and max duration of the utterances are 2.23, 0.09, 18.69 seconds, respectively.

3.2.3 Model configuration

Following the same procedure we used for sequence classification task, we extracted MFCC coefficients over short overlapping frames of size 25 ms and 12.5 ms shift. The number of MFCC vectors will be equal to the number of frames in the audio signal.

Our RNN follows the architecture shown in Figure 4. The RNN model is trained to label each frame of MFCC vector with one of the labels: speech, silence or hesitation. Training takes place over mini-batches, with length of each mini-batch fixed to the longest utterance length along with zero padding. A separate class label is reserved for zero padding frames in the input.

3.3 Initial results

Among the 2884 audio files available, 2500 were used for training and 384 for testing. As for the sequence classification task, we trained the model through 50 epochs, using mini-batches of 32 utterances at a time and a learning rate of 0.05. We used cross-entropy loss function and Adam optimizer. It is taken care that computation of loss function and accuracy are not biased by zero padding frames. The model achieved a train accuracy of 8.24% and a test accuracy of 7.34%. The confusion matrices relative to these accuracy values are shown in Figure 6 and Figure 7. The values reported on the matrices correspond to the number of frames relative to speech, silence and filler tags, in which the audios were originally segmented.

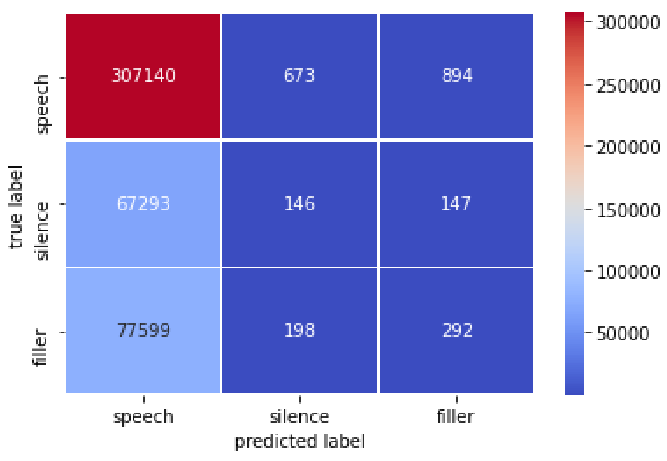

Figure 6: Confusion matrix for the train set.

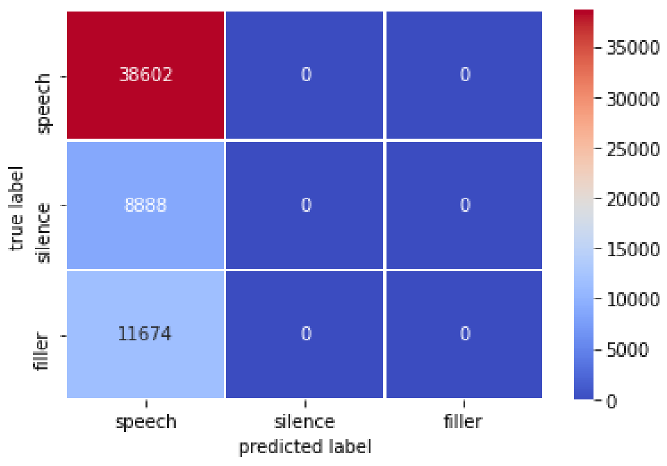

Figure 7: Confusion matrix for the test set.

These poor results are mainly due to the simple RNN architecture that we used for the experiment. An improvement could be achieved augmenting the complexity of the model. In particular, Long Short-Term Memory (LSTM) RNN could be exploited. LSTM are, indeed, more powerful than the basic RNN involved in our experiment. They are able to remember information for a longer period of time and able to use more context to make predictions. Another variant that can be exploited is the Bi-Directional LSTM Network (BLSTM). BLSTM would run the inputs in two ways, one from left to right and one from right to left. Using the two hidden states combined at each frame, it would be possible in any point in time to preserve information from both past and future. Eventually, increasing the numbers of hidden layers and stacking more layers could be another thing to explore.

We did not attempt the aforementioned solutions due to time limitations. In Section 3.3, we will discuss what we learnt from this project, what mistake can be avoided in future research work, and lastly we will discuss our overall experience about having conducted such a project.

Learning and experience

This work on event detection in spontaneous speech has been strongly eventful, so much so that we will no longer undertake a project as we used to. We have learnt various things, from how to approach a research project, to how to organize the team work around it.

Firstly, we have learnt how a little mismanagement of time can unsettle the work flow. We lost a lot of time on tasks that we believed to be crucial, which turned out to be not as significant as we expected. In future work, we will ensure to be more discerning on what is important and what is not for accomplishing the final aim of the project.

Secondly, working on a research project which lasted several months was a premiere for us. It lead us to think broad, as the project could be approached under the angle of our choice. Also, it allowed us to focus our studies on a specific topic over a long period of time, which helped us deepen our understanding of the specific field of speech recognition. Conducting such a long term project resulted in facing delicate situations, such as discouragement when we were clueless, or sense of accomplishment when a problem was successfully solved. Overall, it taught us that anything can be learnt if time and investment are dedicated to the task.

Finally, this project was also an unhoped opportunity to get introduced to the research world. We have got some insights on laboratory and researcher life, which will guide us over our future choice of career. In the same vein, we dived inside the research paper writing task, which will be of crucial help for our future university and professional projects.

Although the results are not the ones we expected, we would like to humbly express the pride we feel to have concluded a project of this importance.

Bibliography

[1] Scikit-learn. URL: `https://scikit-learn.org/stable` (visited on 21/05/2020).

[2] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals. Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):356–370, 2012.

[3] M. Belz and R. Uwe. Pitch characteristics of filled pauses. *The 7th Workshop on Disfluency in Spontaneous Speech (DiSS)*, 2015.

[4] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533, 2012.

[5] P. Boersma and D. Weenink. Praat: doing phonetics by computer. Version 5.3.51. URL: `http://www.praat.org` (visited on 21/05/2020).

[6] R. Brueckner and B. Schulter. Social signal classification using deep BLSTM recurrent neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2014.

[7] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.

[8] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[9] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[10] G. Garg and N. Ward. Detecting filled pauses in tutorial dialogs. *Report of University of Texas at El Paso, El Paso*, 2006.

[11] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.

[12] M. Kaushik, M. Trinkle, and A. Hashemi-Sakhtsari. Automatic detection and removal of disfluencies from spontaneous speech. In *Proceedings of the Australasian International Conference on Speech Science and Technology (SST)*, volume 70, 2010.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] M. Lease, M. Johnson, and E. Charniak. Recognizing disfluencies in conversational speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1566–1573, 2006.

[15] Z. C. Lipton, J. Berkowitz, and C. Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[16] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger. Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Proc. Interspeech 2017*, pages 498–502, 2017.

[17] M. McAuliffe, M. Socolof, E. Stengel-Eskin, S. Mihuc, M. Wagner, and M. Sonderegger. Montreal forced aligner, 2017. Version 1.0.0, retrieved 05 May 2017.

[18] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto. Librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, 2015.

[19] Mozilla. Common voice project official webpage. URL: `https://voice.mozilla.org/en` (visited on 09/05/2020).

[20] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. LibriSpeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[22] R. Qader, G. Lecorvé, D. Lolive, and P. Sébillot. Disfluency insertion for spontaneous tts: Formalization and proof of concept. In *International Conference on Statistical Language and Speech Processing*, pages 32–44. Springer, 2018.

[23] J. Robert. Pydub. URL: `http://pydub.com/` (visited on 22/05/2020).

[24] H. Salamin, A. Polychroniou, and A. Vinciarelli. Automatic detection of laughter and fillers in spontaneous mobile phone conversations. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 4282–4287. IEEE, 2013.

[25] A. Stolcke, E. Shriberg, R. Bates, M. Ostendorf, D. Hakkani, M. Plauche, G. Tur, and Y. Lu. Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Fifth International Conference on Spoken Language Processing*, 1998.

[26] TalkBank. Cabank callhome english corpus. URL: `http://ca.talkbank.org/access/CallHome/eng.html` (visited on 22/05/2020).