Detection of fillers in conversational speech

MSc Natural Language Processing M1 Supervised Project

by

Eduardo Calò, Thibo Rosemplatt

under the guidance of

Imran Sheikh

Academic year 2019-2020

Chapter 1

Introduction

Large vocabulary Automatic Speech Recognition (ASR) systems require a huge amount of human transcribed speech data. Such speech corpora are either purchased from dataset providers or obtained through a data collection process. In either scenario, the costs of acquiring the labelled speech corpora are very high. Recent efforts by the speech research community have led to significantly large amounts of crowdsourced read speech corpora, such as LibriSpeech and Common Voice Project by Mozilla [37, 1], which have been made available publicly at free of cost. With these speech corpora, it is possible to build large vocabulary ASR systems without much or any dependency on expensive speech corpora.

These ASR systems trained on read speech corpora perform well enough under controlled conditions, but they have weaknesses when it comes to dealing with conversational speech. Spontaneous speech is known to be disfluent by nature, containing large amount of fillers, like *"uhm"* and *"ehm"*. Unfortunately, since these ASR systems have never seen such events before in the training corpora, presence of disfluencies can result in a high number of substitutions, insertion and deletion errors, which can seriously hamper the quality of the transcription.

ASR systems must deal with them in order to significantly improve the quality of automatic transcriptions. The best solution would to train a conversational ASR system using only spontaneous conversational speech data. However, obtaining these data is expensive and time-consuming, since large amount of time and labor force are needed for recording and annotating.

Thus, the best arrangement is to automatically detect fillers, given just a small amount of human transcribed dataset. The approaches which have been attempted so far are typically based on sequence classifiers trained on features extracted directly from the speech signal or from the ASR output.

This project will study methods for automatically identifying and annotating fillers in several hours of un-transcribed conversational speech data by training classifiers just using a small amount of human transcribed conversational speech data. Usage of different types of classifiers will be attempted.

Improvement in ASR for conversational speech is the final objective of this project, which is also part of the bigger study on *"Weakly Supervised Learning for Conversational ASR"* under the COMPRISE project [2].

The rest of this report is organised as follows. Section 2 gives a background on Automatic Speech Recognition and conversational speech. This is followed by Section 3 describing the problem targeted by this project. Section 4 discusses the prior work in this area. Finally Section 5 presents the action plan that will be followed by this project.

Chapter 2

Background

2.1 Automatic Speech Recognition

Automatic Speech Recognition is aimed at automatically converting spoken speech into text with the use of a computer program. Denoting the speech input to the ASR as $\mathbf{X}$, the sequence of words at the output of the ASR as $\hat{\mathbf{W}}$, and $\mathbf{W}$ as the possible word sequences to be recognized, ASR can be formulated as:

$$\hat{\mathbf{W}} = \underset{W}{arg\ max}\ P(\mathbf{W})P(\mathbf{X}|\mathbf{W}) \tag{1}$$

where $P(\mathbf{W})$ and $P(\mathbf{X}|\mathbf{W})$ constitute the likelihoods computed by the language modeling and acoustic modeling components, respectively, of the speech-recognition system [22].

A typical ASR architecture is mainly composed of 5 major interrelated sub-components: an acoustic front end, an Acoustic Model (AM), a Language Model (LM), a lexicon and a decoder, as shown in Figure 1 [24].

2.1.1 Acoustic front-end

The first step to perform with an input audio waveform is to extract relevant features that will be used for further analysis. This is what the acoustic front-end exactly does. It converts the speech signal into appropriate features, which provide useful information for recognition. These features are compacted into robust vectors which are later supplied to the recognizer [24].

There are several different feature extraction methods, with the most used being Linear Predictive Coding (LPC) and Mel-Frequency Cepstrum Coefficients (MFCC) [24].

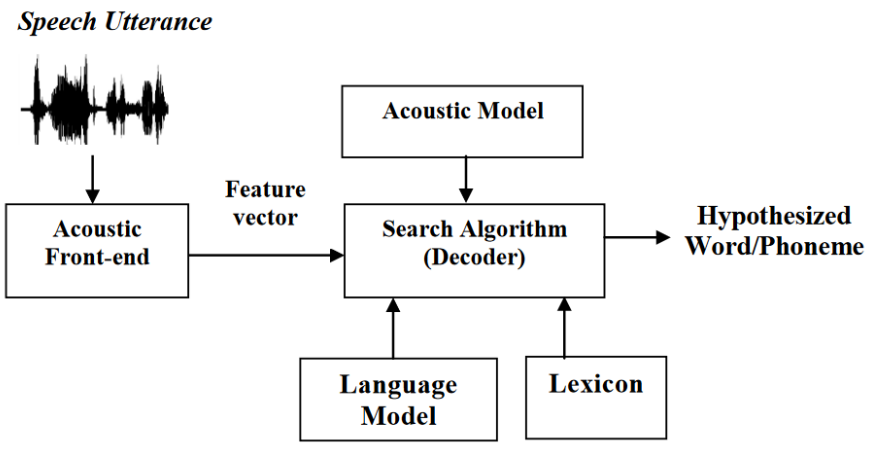

Figure 1: Architecture of a typical ASR system.

Linear Prediction refers to the mechanism of using a linear combination of the signal's past time-domain samples to predict the current time-domain sample [12]. If the prediction is accurate, then a small number of past samples can be used to code a long sequence of the signal. Linear predictive coding of speech signal is based on linear prediction and at the same time tries to model speech as combination of the sound source and a linear acoustic filter [35].

MFCC, on the other hand, is a representation derived from the short term power spectrum of the speech signal, that is mapped into a Mel frequency scale, which closely approximates the human auditory system's response. The first step in MFCC computation is to divide the signal in overlapping time windows ranging between 20 to 40 ms. Shorter time frame would not allow to catch the characteristics needed in the audio signal; longer time frame would make the signal change too much throughout it [21]. In the following step, a Hamming window [24] is applied to each frame, to smooth the sound-wave [32]. After this smoothing, an N-points Fourier Transform is applied on each frame to extract the power spectrum and the constituent frequencies of the signal. Finally, triangular filters (from 20 to 40), imitating the functioning of human's cochlea, are applied. Last step is to take the log of these filters and apply Discrete Cosine Transform to get a compressed representation of the filters. Typically, the resulting 2-13 cepstral coefficients are kept, while the others discarded for ASR [11].

2.1.2 Acoustic Model

The Acoustic Model is used to represent the relationship between an audio signal and the linguistic units (phonemes, syllables, etc.) that make up speech. AM provides the score for $P(\mathbf{X}|\mathbf{W})$ in Equation 1.

Hidden Markov Model (HMM) [6], though old, is still one of the most commonly used backbone of acoustic models. Other type of acoustic models include segmental models [38], supersegmental models including hidden dynamic models [13], neural networks [30], maximum entropy models [26], and (hidden) conditional random fields [20].

HMM can be briefly explained using Figure 2 [42, 40]. Figure 2 shows the first 4 formants of the vowel /a/, divided in 3 segments. It can be seen that the middle portion of the formants does not vary much, compared to the segments on the left and on the right. These segments present more variation, since they are influenced by neighbouring phonemes. So it is convenient to represent the 3 segments with 3 different statistical models. Each such segment is represented by a state in HMM. Figure 3 illustrates the 3 different states in an HMM. So, each spoken word is decomposed into a sequence of basic linguistic units (for example phones), which in turn, have their own HMM, represented by a sequence of states. Individual states of the HMM are typically modelled using a Gaussian Mixture Model (GMM) or a Deep Neural Network (DNN) [40].

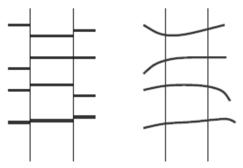

Figure 2: Trajectories of first 4 formants of /a/. On the left, a quasi-stationary approximation of the real formants on the right can be seen.

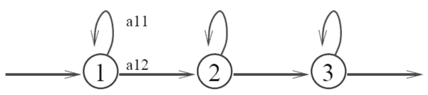

Figure 3: Representation of HMM with 3 states.

2.1.3 Language Model

The role of language modeling in speech recognition is to provide the value $P(\mathbf{W})$ in the fundamental equation of speech recognition, Equation 1 [22]. A language model is a collection of constraints on the sequence of words acceptable in a given language [24].

A basic and widely spread approach for LM, is *n-grams*. It is a stochastic method which describes the probability of the occurrence of a given word, knowing the previous context. The n can take any value ≥ 0, depending on how many words in the previous context are taken into account. For example, if the probability of a word depends purely on the identity of the immediately preceding word, then a 2-gram (bigram) model is used. Similarly, if the two previous words are taken into account in the context, a 3-gram (trigram) model is used. The probability values for each n-gram can be derived from counting its occurrences in the training corpora.

2.1.4 Lexicon

A lexicon is a list of tokens with associated phonetic transcriptions. It connects the tokens that can be recognised by an ASR to the acoustic model. In most cases, different pronunciations are associated to the same token, in order to catch as many nuances as possible from a wide range of speakers. The technique used to achieve this lexicon is called Grapheme-to-Phoneme (G2P) conversion. It is the task of predicting the pronunciation of a word given its graphemic or written form [34]. It can be done by hand or using different algorithms.

2.1.5 Decoder

The decoder merges together all the aforementioned sub-components of an ASR system. The decoding process in a speech recognizer's operation consist of finding a sequence of words which best match the input feature vector sequence [22]. In other words, the task is to find the most likely word sequence $\hat{\mathbf{W}}$ given the observation sequence $\mathbf{X}$, and the acoustic model, lexicon and language model [24]. This task will solve the fundamental equation of ASR seen before (Equation 1). This is a search process, which is usually accomplished using the Viterbi algorithm [45].

2.2 ASR Training

An acoustic model is created from a large speech corpus using special training algorithms to create statistical representations for each linguistic unit. Training is an iterative process. For instance, based on the Maximum Likelihood principle, which means finding the model parameters that maximize the likelihood of the observed data [33]. The training algorithm should be robust enough to generalize to new and unseen samples, which also fits the nature of human speech.

Usually ASR systems are trained using human transcribed speech corpora, which consist of speech recordings collected from human subjects speaking in a pre-defined scenario and audio acquisition setup. Well-known corpora such as Switchboard [17] are available and used for this purpose. However, such speech corpora may have a high price and/or restrictive licenses. Moreover, for some languages, no large transcribed speech corpus exists. It is therefore sometimes necessary to construct a new transcribed speech corpus.

Crowdsourcing platforms are now being used as a new way to collect large transcribed speech corpora, relatively quickly and easily, thanks to inexpensive set-up and possibility for a large number of unskilled people to collect and annotate speech data [23]. An example is the Common Voice Project [1] started by Mozilla in 2017 to create a database for ASR. Another alternative is the LibriSpeech corpus, which consists of about 1000 hours of read English speech made freely available online [37].

However, real-world conversational speech is highly spontaneous. Unlike read speech, it contains pauses, disfluencies, silences and different speech rate and tempo. Since most of these phenomenon are not seen in read speech corpora, ASR trained on read speech corpora perform poorly on real-world conversational speech. In order to address such mismatch in the train and test conditions, ASR adaptation techniques have been proposed. ASR adaptation is performed on new speech datasets covering new tasks or domains. Traditionally, Maximum a posteriori probability estimation and Maximum Likelihood Linear Regression [22] techniques have been used for ASR adaptation. However, more recently, Transfer Learning techniques have been proposed [16].

2.3 Conversational speech and disfluencies

Spontaneous speech is known to be characterised by disfluencies. Disfluencies in speech can be defined as interruptions in normal flow of speech which make utterances longer without adding semantic information [25].

The linguistic phenomenon of disfluency is very common and widespread across all languages. It plays an important role in the structuring of speech and it is even exploited to achieve some linguistic results, such as turn management [7]. Some of them are even considered to be crucial in some pragmatic contexts. For example, a disfluency can arise when a speaker tries to convey a difficult or important message, so that the listener is prepared that some important or difficult matter is about to come and can stay more focused [39]. However, high frequency of disfluencies in speech can be a sign of some sort of speech disorder [10].

Disfluencies can appear under different forms: the most commons are filled pauses (lexicalized and unlexicalized), repetitions, false starts and revisions [43, 36]. Repetitions can be used to gain time and recover the flow of the speech, intensify the effect of an expression, or signal an upcoming problem in the speech [25]. Revisions occur when the speaker slightly fixes his speech after an error. False starts are an extreme case of revisions in which the speaker completely abandons the interrupted speech and starts a fresh one [39].

Unlexicalized filled pauses are vocalizations (phonetically relevant, but semantically void), like *"uhm"*, *"eh"*, *"ah"* etc., which fill the time that should be occupied by a word (generally in correspondence of hesitations, uncertainty or attempts to hold the floor). The duration of vowels in a filled pause is longer than in fluent vowels and have a flat pitch that lies across the median of speaker's pitch. In addition, the energy of the filler is stable and falls towards the end of the utterance. Some useful features for detecting them include duration, pitch, spectral features and formants. In particular, the variation of the first four formant frequencies (F1, F2, F3 and F4) with time can be used as indicators to detect them [25].

Chapter 3

Problems

While ASR models built on crowdsourced read speech corpora can reduce the cost of developing ASR systems, they are unable to handle disfluencies in spontaneous speech. For instance, fillers, like *"uhm"* and *"ehm"*, will be mapped to other words in the lexicon by an ASR trained on read speech. As a result, a sentence spoken as *"how to cook a pizza uhm margherita"*, could be transcribed to *"how to cook a pizza home margherita"*.

Moreover, since the ASR system makes predictions based also on its language model, fillers affect correct recognition of words that are likely to follow each other. As a result, disfluencies result in a high number of substitutions, insertion and deletion errors, which can seriously affect the quality of the transcription [25]. Thus, detecting these events in the speech has the potential to significantly improve the quality of automatic transcriptions.

The optimal solution would be training a conversational ASR system using only spontaneous conversational speech data. However, obtaining such data is not an easy task. First, recording these data incurs in set-up difficulties and finding people willing to perform this task. After recording process, manual transcriptions of the audio recordings must be performed, which is also expensive and time consuming.

In order to factor out the detrimental effect of fillers, partially avoiding labor intensive, expensive, time-consuming and error-prone tasks such as collecting speech samples and annotating them, the best trade-off is to automatically identify fillers in several hours of untranscribed spontaneous speech by training classifiers just using a small amount of human transcribed conversational speech data. This project will explore methods for automatically identifying and annotating fillers in spontaneous speech with an objective to improve the quality of automatic transcriptions for training spontaneous speech ASR.

Chapter 4

Prior art

A number of different techniques have been proposed so far to detect fillers in speech signals. These approaches can be categorized into 3 main groups: detection using prosodic features, detection using lexical information, and detection using spectral information.

4.1 Detecting fillers using prosodic features

This approach is purely based on using acoustic prosodic features (i.e. pitch, duration, and energy of the voice) to detect fillers.

Garg et al. propose an approach based only on prosody [15]. They try to detect fillers automatically, given an audio signal using the following assumptions:

- The pitch of fillers lies along the median of the pitch of the user across all his utterances
- A filler has a flat pitch
- The energy of a filler is stable or falling towards the end of their utterance

Although their approach works, it yields poor results for both coverage (proportion of the actual fillers that were found) and accuracy (proportion among the detected segments that were actual fillers). The errors raised can be sorted into two types: false negatives (they occur when an instance is classified as non-filler, whereas it actually is) and false positives (as opposition, they occur when an instance is classified as being a filler, whereas it is actually not). False negatives can be explained because the pitch of the fillers has been finally found out not to be as flat as expected. Also, the duration of the fillers was often too small compared to what was expected. On the other hand, false positives have occurred because of some long vowels at the beginning or at the end of the word have been classified as fillers.

In another study by Kaushik et al. [25], filler detection is based on the variation of prosody (f0), and formants (F1, F2, F3, F4). Their hypothesis was that the pitch of a filler is stable and lies around the median of speaker pitch, and similarly that formants are stable as well. Stability of formants at a given voiced segment was measured by computing the Standard Deviation (SD) of each formant frequency within each frame. The accuracy was not high enough, and the deletions, substitution, and insertion errors were too frequent.

Some other studies related to prosodic features of the filler have been conducted, even though they are not specifically focused on how to detect a filler in a speech utterance. For example, Belz et al. [7] have found that filled pauses are more likely to appear in monomodal condition (with no visual cue, e.g., in a phone conversation). This can help in order to know in what context fillers must be expected the most. Also, they have found that a filler either having a rising or falling f0 contour indicates that it is meant for holding the turn of the conversation. Conversely, a filler having a more flat contour is more the consequence of a hesitation.

4.2 Detecting fillers using LM information

Another approach, which has been attempted, takes advantage of the information provided by the LMs. In [41], detection, classification and segmentation of 4 different types of interval (fillers, speech, laughter, silence) in phone conversations is tried with the aid of a 2-gram statistical LM to estimate a-priori the probability that one category follows the other (like, e.g., it is more likely that a filler follows speech, rather than laughter). The results show that such language models can significantly improve the performance of purely acoustic feature based approach.

In [44], a combined approach for the detection of inter-word events (sentence boundaries and four classes of disfluencies, including fillers) on spontaneous speech transcribed by an automatic recognizer is also tried. The system combines prosodic and Language Model sources. The authors claim that the results reported in this paper could be further improved by including POS tagging or other syntactic information.

Another approach, slightly different, but using the aid of LMs, is proposed in [39]. This implementation relies on conditional random fields (CRFs) and LMs to demonstrate the possibility to include fillers in written texts, producing plausible disfluent utterances. This is an innovative formalization of a theoretical process of disfluency composition and insertion in texts. The experiments conducted on this implementation show that the proposed process is functional, although requiring further improvements.

Lease et al. in [28] present, instead, a system for modeling disfluency in conversational speech: fillers, repairs, and Interruption Points (IPs). For each sentence, candidate repair analyses are generated by a stochastic Tree Adjoining Grammar (TAG) noisy-channel model. A probabilistic syntactic language model scores the fluency of each analysis, and a maximum-entropy model selects the most likely analysis given the language model score and other features. Fillers are detected independently via a small set of deterministic rules, and IPs are detected by combining the output of repair and filler detection modules.

4.3 Detecting fillers using spectral information

So far the most successful approach is the one proposed by Brueckner et al. [9]. In this study, the effect of applying deep Bi-directional Long-Short-Term Memory (BLSTM) Recurrent Neural Networks for detection and classification of fillers and laughter is investigated. The task was to perform a frame-wise [19] classification of phone-call recordings into three vocalization classes, including fillers. Using open-source feature extractor tool openSMILE (covered in Section 5.2.2), frame-wise low-level descriptors (LLDs) and functionals were extracted every 10 ms adopting a frame size of 25 ms. As typical in ASR, frame-wise logarithmic energy and Mel-Frequency Cepstral Coefficients (MFCC) 1–12 were computed (see Section 2.1.1). After trying several different neural network architectures with various combinations, they found that deep BLSTM models (created by stacking several BLSTMs and by combining non-recurrent deep neural networks with BLSTMs) is the most successful.

Chapter 5

Action plan

In this section an attempt of the implementation plan that will be followed to achieve the goals of the project will be presented, viz. automatic detection of fillers in conversational speech. First, a note on the approaches that will be tried is presented. This includes the features extracted from speech and the detection methods. This is followed by the choice of software tools for speech feature extraction and for developing the detection methods. Finally, the timeline planned for implementation of this project will be presented.

5.1 Approach for detection of filler events in speech

5.1.1 Speech features

MFCC

Relevant features that characterize the input speech will be extracted through Mel Frequencies Cepestral Coefficient (MFCC) technique. MFCC has been discussed in Section 2.1.1.

Other audio descriptors

Apart from MFCC, different Low Level audio Descriptors (LLD) will also be explored as well as High Level audio Descriptors (HLD), which are commonly used in other music and speech interpretation tasks [14]. LLD includes features such as pitch, formants, voice quality features, as well as LPC. HLD typically include functionals like mean, moments, percentiles, etc. calculated over the LLD.

5.1.2 Detection and Classification

In this project, different classifying methods will be attempted.

Sliding window approach

The basic concept of this classifying method is pretty transparent. The signal will be processed segment by segment. Each segment is called window. Once processed, the window will slightly shift on the right such that all the windows processed overlap. Each window processed is then classified as a filler, or any other speech sound (see Figure 4) [29]. The length and shift of the window are important, and they could influence fillers detection results.

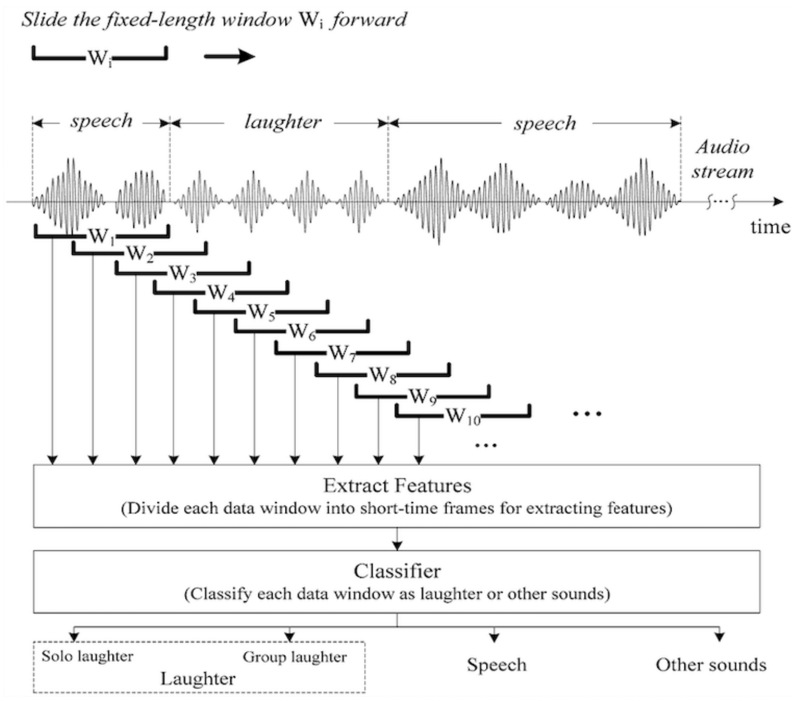

Figure 4: Sliding window approach.

Sequence based classifier

A speech signal can also be classified using a Recurrent Neural Network (RNN) [18]. In such networks, the output of a hidden layer depends on the output of the previous layer. For example, if one wants to predict the price of a stock at a given time, or wants to predict the next word in a sequence, it is imperative that dependence on previous observations is considered. RNN have indeed a loop. To this extent, one can state that these kinds of networks have a memory.

There exists a variant of RNN, which is even more powerful, and is able to use more context to make predictions: the Long-Short-Term-Memory Network [8]. LSTMs are explicitly designed to avoid the long sequence training problems of RNNs. Remembering information for long periods of time is practically their default behavior. Bi-Directional Long Short Term Memory Network (BLSTM) are a variant of LSTM. Unidirectional LSTM only preserves information of the past, because the only inputs it has seen are from the past. Using BLSTM will run the inputs in two ways, one from past to future and one from future to past. What differs this approach from unidirectional is that in the LSTM that runs backwards you preserve information from the future and using the two hidden states combined you are able in any point in time to preserve information from both past and future.

5.2 Tools

In the following sections, tools planned to be used in this project will be listed.

5.2.1 LibROSA

LibROSA [31, 3] is a powerful Python library used for audio analysis. It will be used to extract MFFC features from the speech signal.

5.2.2 OpenSMILE

OpenSMILE is an open source software, whose main use is to automatically extract features from a speech signal, and to classify them [4]. SMILE stands for *"Speech and Music Interpretation by Large space Extraction"* [14]. It will be used to extract the LLD and HLD features mentioned in Section 5.1.1.

5.2.3 Scikit-learn

Scikit-learn [5] is one of the most recognized and used Python library for machine learning tasks. It is based on the SciPy library, as suggested by its original name *"SciPy Toolkit"*. This library will be used to develop the classification and detection methods discussed in Section 5.1.2.

5.3 Corpora

Our experiments will be conducted using mainly two corpora: the English subset of the VERBMOBIL corpus of human dialogs [27] and the LibriSpeech corpus of read English speech [37]. The English subset of the VERBMOBIL corpus consists of about 30 hours of human transcribed speech. This corpus will be divided into 3 subsets: one for training (filler detectors, classifiers and ASR systems), one for validation and one used as test set. Automatic transcriptions of VERBMOBIL corpus will be prepared using an English ASR pre-trained on 1000 hours of read speech from the LibriSpeech corpus.

5.4 Timeline

The project is planned to progress through the following stages, during the period January - May 2020.

- Month 1 (Jan 2020)
  - Introduction to tools and corpora discussed in Section 5.2 and Section 5.3 respectively.
  - Analysis of the errors obtained by ASR trained on LibriSpeech in filler regions of the utterances from VERBMOBIL corpus.
- Month 2-3 (Feb - Mar 2020)
  - Experiments on filler detection based on features extracted directly from the speech signal.
  - Exploration of different features, different models as well as model combinations.
- Month 4-5 (Apr - May 2020)
  - Analysis on the (minimum) amount of annotated conversational speech data required for effective learning of fillers.
  - Improvements in ASR trained on automatic transcriptions of VERBMOBIL corpus, post proposed filler detection approach.
  - Compilation of results, final report and defense preparations.

Bibliography

[1] Common voice project official webpage. `https://voice.mozilla.org/en`.

[2] Comprise. `https://www.compriseh2020.eu`.

[3] Librosa documentation. `https://librosa.github.io/librosa/`.

[4] Opensmile official page. `https://www.audeering.com/opensmile/`.

[5] Scikit-learn official page. `https://scikit-learn.org/stable`.

[6] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3(1):1–8, 1972.

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

[8] J. Brownlee. *Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning*. Machine Learning Mastery, 2017.

[9] 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.

[10] A. K. Cordes. The reliability of observational data: I. theories and methods for speech-language pathology. *Journal of Speech, Language, and Hearing Research*, 37(2):264–278, 1994.

[11] 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.

[12] L. Deng and D. O'Shaughnessy. *Speech processing: a dynamic and optimization-oriented approach.* CRC Press, 2018.

[13] L. Deng, D. Yu, and A. Acero. Structured speech modeling. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1492–1504, 2006.

[14] F. Eyben, M. Wöllmer, and B. Schuller. Opensmile: The munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 1459–1462, New York, NY, USA, 2010. ACM.

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

[16] P. Ghahremani, V. Manohar, H. Hadian, D. Povey, and S. Khudanpur. Investigation of transfer learning for asr using lf-mmi trained neural networks. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 279–286, Dec 2017.

[17] J. J. Godfrey and E. Holliman. Switchboard-1 release 2. *Linguistic Data Consortium, Philadelphia*, 926:927, 1997.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[19] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, July 2005.

[20] A. Gunawardana and W. Byrne. Discriminative speaker adaptation with conditional maximum likelihood linear regression. In *Seventh European Conference on Speech Communication and Technology*, 2001.

[21] X. Huang, A. Acero, H.-W. Hon, and R. Foreword By-Reddy. *Spoken language processing: A guide to theory, algorithm, and system development.* Prentice hall PTR, 2001.

[22] X. Huang and L. Deng. An overview of modern speech recognition. In *Handbook of Natural Language Processing, Second Edition*, 2010.

[23] T. Hughes, K. Nakajima, L. Ha, A. Vasu, P. Moreno, and M. Lebeau. Building transcribed speech corpora quickly and cheaply for many languages. In *Interspeech*, pages 1914–1917, 2010.

[24] S. Karpagavalli and E. Chandra. A review on automatic speech recognition architecture and approaches. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 9(4):393–404, 2016.

[25] 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.

[26] H.-K. J. Kuo and Y. Gao. Maximum entropy direct models for speech recognition. *IEEE Transactions on audio, speech, and language processing*, 14(3):873–881, 2006.

[27] A. Kurematsu, Y. Akegami, S. Burger, S. Jekat, B. Lause, V. L. Maclaren, D. Oppermann, and T. Schultz. Verbmobil dialogues: Multifaced analysis. In *Sixth International Conference on Spoken Language Processing*, 2000.

[28] 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.

[29] Y. Li and Q. He. Detecting laughter in spontaneous speech by constructing laughter bouts. *International Journal of Speech Technology*, 14:211–225, 09 2011.

[30] R. P. Lippmann. An introduction to computing with neural nets. *IEEE Assp magazine*, 4(2):4–22, 1987.

[31] B. McFee, C. Raffel, D. Liang, D. 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*, pages 18–24, 01 2015.

[32] A. R. Mohamed. *Deep neural network acoustic models for ASR.* PhD thesis, University of Toronto, 2014. `https://tspace.library.utoronto.ca/bitstream/1807/44123/1/Mohamed_Abdel-rahman_201406_PhD_thesis.pdf`.

[33] N. M. Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[34] J. R. Novak, N. Minematsu, and K. Hirose. Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the wfst framework. *Natural Language Engineering*, 22(6):907–938, 2016.

[35] D. O'Shaughnessy. Linear predictive coding. *IEEE potentials*, 7(1):29–32, 1988.

[36] D. O'Shaughnessy. Recognition of hesitations in spontaneous speech. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 521–524. IEEE, 1992.

[37] 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.

[38] A. B. Poritz. Hidden markov models: A guided tour. In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7–13, 1988.

[39] 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.

[40] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[41] 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.

[42] K. Samudravijaya. Speech and speaker recognition: A tutorial. In *Proc. Int. Workshop on Tech. Development in Indian Languages, Kolkata*, 2003.

[43] E. Shriberg. To 'errrr' is human: ecology and acoustics of speech disfluencies. *Journal of the International Phonetic Association*, 31(1):153–169, 2001.

[44] 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.

[45] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.