MASTER THESIS

Vojtěch Hudeček

Exploiting user's feedback to improve pronunciation of TTS systems

Institute of Formal and Applied Linguistics
Faculty of Mathematics and Physics
Charles University in Prague

Supervisor of the master thesis:	doc. Ing. Zdeněk Žabokrtský, Ph.D.
Study programme:	Informatics
Study branch:	Artificial Intelligence

Prague 2017

Title: Exploiting user’s feedback to improve pronunciation of TTS systems

Author: Bc. Vojtěch Hudeček

Department: Name of the department

Supervisor: doc. Ing. Zdeněk Žabokrtský, Ph.D, Institute of Formal and Applied Linguistics

Abstract: Although spoken dialogue systems have greatly improved, they still cannot handle communications involving unknown topics and are very fragile. We will investigate methods that can improve spoken dialogue systems by correcting or even learn the pronunciation of unknown words. Thus we will provide better user experience, since for example mispronounced proper nouns are highly undesirable. Incorrect pronunciation is caused by imperfect phonetic representation, typically phonetic dictionary. We aim to detect incorrectly pronounced words by exploiting userâĂŹs feedback as well as using prior knowledge of the pronunciation and correct the transcriptions accordingly. Furthermore, the learned phonetic transcriptions can be used to improve speech recognition module by refining its models. Models used in speech recognition cannot handle words that are not in their vocabulary or have phonetic representation. Extracting those words from userâĂŹs utterances and adding them to the vocabulary should lead to a better overall performance.

Keywords: text-to-speech, automatic speech recognition, user’s response, phonetic dictionary, machine learning, mel cepstral distortion

Dedication.

Contents

Introduction

1. Introduction

1.1 Introduction to the problematic

Voice control or communication is a common feature of many systems nowadays. Its applications ranges from simple one-word control commands to complex communication in spoken dialogue systems. In this work, we consider mainly these complex systems. For the sake of clarity, we now briefly describe setting of such system. It usually contains Automatic Speech Recognition (ASR) module, so the natural speech can be recognized and translated into words. The system then somehow derives an appropriate response, typically in the form of sentence written in natural language. This response can be displayed in the textual form, however, it is more common to generate an audio with human voice reading the response. Although it is possible to use a set of prerecorded utterances, this approach has obvious limitations since it is not able to read an arbitrary phrase. Particularly, it may be difficult to read named entities and numerical values such as time and date. Also, the usage of variable utterances provides better user experience. Because of this, a Text-To-Speech (TTS) module is usually also part of dialogue systems. The purpose of this module is to transform a (generally arbitrary) written text utterance to natural speech. Modern TTS systems produce audio waveforms that sound quite naturally and the pronunciation is sufficiently good. Nevertheless, it may experience some difficulties, mainly when it comes to unknown words. This may happen, because the system is usually trained using certain set of words, typically from one language. But real applications often require to pronounce named entities or other language- or domain- specific words, that cannot be present during the training phase. This cause situations, when the system has to emply some (imperfect) mechanism to derive the pronunciation and the words may be mispronounced. Succh words are called Out-Of-Vocabulary (OOV). Although the does not occur often, the negative effect can be quite strong, since it is inconvenient for the user when his or her name is pronounced with mistakes.

In this work, we aim to improve the TTS system pronunciation of OOV words. First, we explore method that can identify words, that are potentially difficult to pronounce. The identification is first step towards the improvement. We propose several measures that can reflect badly pronounced words without any prior language knowledge. To achieve this, we employ the user and obtain correct pronunciations from him. So we get training examples and we are able to improve the TTS system by processing the obtained recording, deriving a phonetic transcription (i.e. pronunciation) and adding it to the TTS vocabulary. Moreover, the derived pronunciations can be used to improve the recognition ability of the ASR module, since it is also dictionary-based. As it has been suggested, our method has got potentially very useful applications. It can be used to enlarge vocabularies of TTS or ASR systems both offline or on the fly using the user's feedback. There exist several ways how to obtain such a feedback, however, this is not a subject of this work. Theoretically, the method can work with just one gold example, however, it is better to obtain more recordings in general.7 In dialogsample we provide basic example of simple dialogue, illustrating how real application could look like. However, in this work we assume the user's recording(s) have been gathered already and we do not consider the dialogue policy.

```
System:  Hello, /AANDRZHEZH/.
User:  You said it wrong, my name is /ONDRZHEI/.
System:  /ANDREY/, correct?
User:  No, it is /ONDRZHEI/.
System:  Oh, /ONDRZHEI/?
User:  That's right.
```

Figure 1.1: Sample dialogue illustrating the pronunciation correction. The transcriptions of the user's name are given in ARPABETArp

1.2 Overview of used techniques

1.2.1 Audio signal processingTaylor [2009]

Details of this part are beyond the scope of this book, however we sketch here the basic principles, so we can descrbie our input data. Speech signal in the real world are mechanical waves, so it is obviously analogous quantity. When we process the speech signal with computers, it is assumed to be digitised, so it is converted into discrete form. When performing digital signal processing, we are usually concerned with three key issues.

1. To remove the influence of phase
2. Performing source/filter separation, so that we can study the *spectral envelope* of sounds. Spectral envelope characterizes the frequency spectrum of the signal, which is essential for the speech recogniton and processing.
3. We often wish to transform these spectral envelopes and source signals into more efficient representations.

Overview of the main processing steps follows. The input signal is divided into *windows*. Windowing considers only some part of the signal. Usually, the signal is transformed at window borders to prevent discontinuities. This is achieved for example by use of *Hamming* window. The complete waveform is therefore a series of windows, that are sometimes called *frames*. The frames overlap, this is influenced by the size of the *frame shift*. Inside one frame, we assume, that the speech signal is stationary and we transform it to a frequency domain by Discrete Fourier Transformation[1]. It is better for observing the natural speech characteristics, to use a logarithm scale instead of linear. In fact, the *mel scale*[2] is frequently used, which even better corresponds with the human perception. Further, the so called *cepstrum*[3] is computed from the log magnitude spectrum, by performing inverse Fourier transformation. Cepstrum can be

[1]https://en.wikipedia.org/wiki/Discrete_Fourier_transform
[2]https://en.wikipedia.org/wiki/Mel_scale
[3]https://en.wikipedia.org/wiki/Cepstrum

represented by coefficients, number of which can be chosen. Those are called the *Mel Frequency Cepstral Coefficients.* To sum up, we have described how to process the digital audio signal and convert it into discrete series of overlaping frames, each of which is represented by a fixed-length vector of MFCCs.

We gave a brief overview of the signal processing procedure that is essential to work with the audio signal. When we mention the input speech signal, we mean series of MFCC vectors, unless explicitly stated otherwise.

1.2.2 Finite state transducersMohri [1997]

Finite state transducers (FSTs) are finite state automata that are augmented by addition of output labels to each transition. We can further modify the FST and add a weight to each edge. We define a semiring $(S, \oplus, \otimes, \overline{0}, \overline{1})$ as a system that fullfills the following conditions:

- $(S, \oplus, \overline{0})$ is a commutative monoid with identity element $\overline{0}$
- $(S, \otimes, \overline{1})$ is a monoid with identity element $\overline{0}$
- $\otimes$ distributes over $\oplus$
- $\forall a \in S : a \otimes \overline{0} = \overline{0} \otimes a = \overline{0}$

We this definition of the semiring, we can describe a weighted FSTMohri [2009] T over a semiring S formally as a 8-tuple $(\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ where Σ, Δ are finite input and output alphabets, Q is a finite set of states, $I \subset Q$ is a set of initial states, $F \subset Q$ is a set of final states, E is a multiset of transitions, $\lambda : I \to S$ is an initial weight function and $\rho : F \to S$ is a final weight function. Each transition is element of $Q \times \Sigma \times \Delta \times S \times Q$. Note, that since E is a multiset, it allows two transitions between states p and q with the same input and output label, and even the same weight. However, this is not used in practice. An example of weighted FST is given in TODO. Finite state transducers have been used widely in many areas, especially linguistics. They can represent local phenomena encountered in the study of language and they usage often leads to compact represantations. Moreover, it is also very good from the computational point of view, since it advantageous in terms of time and space efficiency. Whole area of algorithms has been described that consider FSTs. One of the most important is *determinization*, which determinize paths in the FST and thus allows the time complexity to be linear in the length of input. Weighted FSTs have been widely used in the area of automatic speech recogniton. The allows to compactly represent the hypothesis state of the acoustic model as well as combining it with the information contained in the language model.

1.2.3 Automatic speech recognition (ASR)

Overview

The task in ASR is quite clear: An utterance spoken in natural language should be translated into its text representation. Formally, we are given a sequence of acoustic observations $\mathbf{X} = X_1 X_2 \dots X_n$ and we want to find out the corresponding word sequence $\mathbf{W} = W_1 W_2 \ldots W_m$ that has a maximum posterior probability $P(W|X)$i.e., according to the Bayes rule:

$$\hat{\mathbf{W}} = argmax_w \frac{P(\mathbf{W})P(\mathbf{W}|\mathbf{X})}{P(\mathbf{X})} \tag{1.1}$$

The challenge consists of two parts. First, we have to build accurate acoustic model that describes the conditional distribution $P(W|X)$. Second part is to create a language model that reflects the spoken language that should be recognized. Usually, a variation of the standard N-gram approach is sufficient. Individual words are composed of phonetic units, which are in turn modeled using states in probabilistic machinery such as a finite state transducers. Because the speech signal is continuous, it is transformed to discrete sequence of samples. MFFC's are commonly used to represent the signal. The process of deriving this sequence is described in further detail in the 1.2.1. In general, it is difficult to use whole-word models for the acoustic part, because every new task may contain unseen words. Even if we had sufficient number of examples to cover all the words, the data would be too large. Thus, we have to select basic units to represent salient acoustic and phonetic information. These units should be *accurate*, *trainable* and *generalizable*. It turns out that the most appropriate units are phones. Phones are very good for training and generalization, however, it does not include contextual information. Because of this, so called triphones are commonly used. To model the acoustic, Hidden Markov Models are commonly used, because it can deal with unknown alignments. In recent years, neural networks have experienced big breakthrough and it found application even in the area of speech recognition. Namely, recurrent neural networks are able to work with an abstraction of memory and process sequences of variable length, so it overcomes the HMM's in terms of accuracy. The language and acoustic models are traditionally trained independently and they are joined during the decoding process. The decoding process of finding the best matched word sequence **W** to match the input speech signal **X** in speech recognition systems is more than a simple pattern recognition problem, since there is an infinite number of word patterns to search in continuous speech recognition. A FST is built in order to make the decoding. Its nodes represent acoustic units and edges are assigned costs. This assignment corresponds to the likelihood determined by the acoustic model as well as the language model, so the FST is a composition of theses two. Decoding the output is realiztheed as searching the best path through this graph.

The decoding graph is costructed in such a way, that the path can contain only words present in the vocabulary the language model is built on. Thus, each possible path consists of valid words. Because of that, we can see the word as a basic unsplittable unit, although it is actually composed from phones, or triphones respectively. However, we can build a FST that allows to construct the path from phones and thus recognize the input on the phonetic level. Also the search algorithm preserves alternative paths so in the end it gives us not only the best path but also list of alternative hypotheses. We call it the n-best list. Each hypothesis in the n-best list is associated with its likelihood that expresses information from both the language and acoustic model. In chapter ?? we explore the n-best list and propose method that exploits it.

1.2.4 Text-to-speech (TTS)

OverviewTaylor [2009]

The text-to-speech problem can be looked at as a task of transforming an arbitrary utterance in natural language from its written form to the spoken one. In general, these two forms have commonalities in the sense, that if we can decode the form from the written signal, then we have virtually all the information we require to generate a spoken signal. Importantly, it is not necessary to (fully) uncover the meaning of the written signal, i.e. employ Spoken Language Understanding (SLU). On the other hand, we may need to generate prosody which adds some information about emotional state of the speaker or emphasizes certain parts of the sentence and thus changing the meaning slightly. To obtain prosodic information, sophisticated techniques need to be involved, since its not a part of common written text, except some punctuation. Another difficulties stems from the fact, that we often need to read numbers, or characters with special meanings such as dates or mathematical equations. The problem of converting text into speech has been heavily explored in the past and the TTS systems (engines) are very sophisticated nowadays. The subsequent section, describes briefly the typical architecture of TTS system.

TTS system architectureTaylor [2009]

Let us now describe a common use of TTS system. The architecture usually consists of several modules, although some end-to-end systems base on neural networks have appeared recentlyvan den Oord et al. [2016] Wang et al. [2017]. First, the input text is divided into sentences and each sentence is further tokenized, based on whitespace characters, punctuation etc. Then, the non-natural language tokens are decoded and transformed into text form. We then try to get rid of ambiguity and do prosodic analysis. Although much of the information we might ideally like is missing from the text, we use algorithms to determine the phrasing, prominence patterns and tuning the intonation of the utterance. Next is the synthesis phase. The first stage in the synthesis phase is to take the words we have just found and encode them as phonemes. There are two main approaches how to deal with the synthesis phase. More traditional approach is so called **unit concatenation**. This approach uses a database of short prerecorded segments of speech (roughly 3 segments per phoneme). These segments are then concatenated together with some use of signal processing so they fit together well. An alternative is to use statistical, machine learning techniques to infer the specification-to-parameter mapping from data. While this and the concatenative approach can both be described as data-driven, in the concatenative approach we are effectively memorizing the data, whereas in the statistical approach we are attempting to learn the general properties of the data. Trained models are able to transform the phonemes into audio signal representation. This representation is then synthesized into waveforms using a vocoder module. Although unit concatenation approaches generally achieve better results, the statistical ones are more flexible and have good possibilities to fine tuning or postprocessing.

Grapheme to Phoneme conversion and its drawbacks

In the stage of converting graphemes (i.e. text) to phonemes, the g2p module is usually used. The g2p task is to derive pronunciations from ortographic transcription. Traditionally, it was solved using decision trees models. It can also be formulated as a problem of translating one sequence to another, so neural networks can be used to solve this taskYao and Zweig [2015]. However, in our work we use joint-sequence modelBisani and Ney [2008], which estimates joint distribution of phonemes based on probabilities derived from n-grams. The g2p module is crucial component of the system and it has great impact on the final pronunciation. Since we use machine learning techniques and train on the dictionary data, the model inherently adopts pronunciation rules from the respective language. Although it is in principle possible to train multilingual g2pSchlippe et al. [2012], it is usually not the case in common TTS systems. The problem arises, when words that originate in some foreign language should be pronounced. In this work we address this issue by deriving pronunciations from the ASR output.

Speech Synthesis Markup Language (SSML)Taylor and Isard [1997]

SSML is a standard, that allows to specify aspects of the speech. It is an input to the synthesis module and many TTS engines support its use. We can specify emotions, breaks etc. with the help of SSML. More importantly, it can be used to input particular phonemes and thus circumvent the g2p module. In our work we use this method to feed our phonetic transcriptions into the engine. The disadvantage of this method is, that many TTS engines process the SSML input imperfectly or not at all. However, in principle it is possible to add the transcriptions directly into the system's vocabulary.

Overview of the used TTS systems

1. Cereproc TODO:citations This engine is a commercial software that is free for educational purposes. Although it is not open-sourced, we use it for its good quality and reliable outputs.

2. MaryTTS MaryTTS is a German open-source project written in Java. It is highly customizable since it is modular. Thus we can explore output of an arbitrary module or replace it in the processing pipeline. MaryTTS is based on client-server architecture.

3. gTTS is a TTS service provided online by Google. It achieves very good quality, however it allows ony use of one voice in the free version.

4. Pico SVOX Pico TTS is a lightweight engine that lacks a good quality of the gTTS, however it offers a large selection of voices and it is commonly used on the phones with Google's Android operating system.

1.2.5 Algorithms description

Dynamic Time Warping

Ratanamahatana and Keogh [2004] The measurement of similarity between two time series is an important subroutine in many applications. Moreover, sometimes we need to align two sequences that describe same data but are of different lengths. Both this tasks can be solved with use of DTW. Suppose we have two time series, a sequence Q of length n, and a sequence C of length m, where

$$Q = q_1, q_2, \ldots, q_i, \ldots, q_n$$
$$C = c_1, c_2, \ldots, c_j \ldots, c_m$$

To align these two sequences using DTW, we first construct an n-by-m matrix where the (i^{th}, j^{th}) element of the matrix corresponds to the squared distance, $d(q_i, c_j) = (q_i$âĂŞ$c_j)$, which is the alignment between points q_i and c_j. To find the best match between these two sequences, we retrieve a path through the matrix that minimizes the total cumulative distance between them. In particular, the optimal path is the path that minimizes the warping cost:

$$DTW(Q, C) = \sqrt{\sum_{k=1}^{K} w_k}$$

where w_k is the matrix element $(i, j)_k$ that also belongs to k^{th} element of a warping path W, a contiguous set of matrix elements that represent a mapping between Q and C. Methods of dynamic programming are used to fill in the values and find alignment of sequences. Thus a result of this algorithm is a mapping σ that can be used to construct sequence of pairs A containing members of both sequences C, Q in each time step. σ is constructed in such a way, that it holds:

$$A_i = (C_{\sigma(C,i)}, Q_{\sigma(Q,i)})$$
$$i = 1 \ldots K$$
$$max(|C|, |Q|) \leq K$$
$$\text{and } \sum_{i=0}^{K} d(\sigma(C, i), \sigma(Q, i) \text{ is minimal}$$

Where $d(x, y)$ is an arbitrary distance metric. In our application, we want to align two sequences of phonemes, so we choose Levensthein distance to be our metric.

1.3 Related Work

1.3.1 Grapheme-to-phoneme conversion

An automatic grapheme-to-phoneme conversion was first considered in the context of TTS applications. The input text needs to be converted to a sequence of phonemes which is then fed into a speech synthesizer. It is common in TTS systems that they first try to find the desired word in the dictionary and if it doesn't find it, it employs the grapheme-to-phoneme ($g2p$) module. A trivial approach is to employ a *dictionary look-up*. However, it cannot handle context and inherently covers only finite set of combinations. To overcome this limitations, the rule-based conversion was developed. Kaplan and Kay Kaplan and Kay [1994] formulate these rules in terms of finite-state automata. This system allows to greatly improve coverage. However the process of designing sufficient set of rules is difficult, mainly since it must capture irregularities. Because of this, a *data-driven* approach based on machine learning has to be employed. Many such techniques were explored, starting with Sejnowski and Rosenberg Sejnowski and Rosenberg [1988]. The approaches can be divided into three groups.

Techniques based on local similarities

The techniques presuppose an alignment of the training data between letters and phonemes or create such an alignment in a separate preprocessing step. The alignment is typically construed so that each alignment item comprises exactly one letter. Each slot is then classified (using its context) and a correct phoneme is predicted. Neural networks and decision tree classifiers are commonly used for this task.

Pronunciation by analogy

This term is typically used for methods that could be described as nearest-neighbor-like. They search for local similarities in the training lexicon and the output pronunciation is chosen to be analogous to retrieved examples. In the work of Dedina and Nusbaum [1991] the authors build a pronunciation lattice from the words that match the input string. Paths through the lattice then represent the derived pronunciations. This approach has been further improved.

Probabilistic approaches

The problem can also be viewed from a probabilistic perspective. Pioneers in this area were Lucassen and Mercer [1984] They create 1-to-n alignments of the training data using a context independent channel model. The prediction of the next phoneme is based on a symmetric window of letters and left-sided window of phonemes. Authors then construct regression tree,the leafs of which carry probability distribution over the phonemes. Other approaches may be based on statistical parsing

In the work of McGraw et al. [2013], the concept of Pronunciation Mixture Models is introduced. Authors use a special kind of speech recognizer, search space of which has four primary hierarchical components: the language model G, the phoneme lexicon L, the phonological rules P that expand the phoneme pronunciations to their phone variations, and the mapping from phone sequences to context-dependent model labels C. These can be with advantage represented as FSTs and thus the full decoder network can be represented as a composition of these components: $R = C \circ P \circ L \circ G$. A probabilistic lexicon is considered in a sense, that several pronunciations are allowed for each word and there is no hard limitation that would force the recognzier to choose one. Instead, kind of soft voting is considered, meaning, that each transcription is used with certain probability. Also, joint-sequence modelling is considered, as we introduced in **??**, so each transcription is considered together with its orotgraphic form. That means, that we can describe the log-likehood of M utterances $D = \{\mathbf{u}_i, \mathbf{W}_i\}$ where $\mathbf{u}_i$ are speech data and $\mathbf{W}_i$ their transctiptions as follows:

$$\mathcal{L}(\Theta|D) = \sum_{i=1}^{M} log \sum_{B \in \mathcal{B}} P(\mathbf{u}_i, \mathbf{B}, \mathbf{W}_i; \Theta) \tag{1.2}$$

where $\mathbf{B}$ are respective phone sequences (i.e. pronunciations) and Θ represents the model parameters. Then, we derive using a chain rule:

$$P(\mathbf{u}_i, \mathbf{B}, \mathbf{W}_i; \Theta) = P(\mathbf{u}_i|\mathbf{B})P(\mathbf{B}|\mathbf{W}_i; \Theta)P(\mathbf{W}_i) \tag{1.3}$$

If we further assume, that pronunciation sub-units $\mathbf{b}_i \in \mathbf{B}$ are context independent, we can transcribe the above expression:

$$P(\mathbf{u}_i|\mathbf{B})P(\mathbf{B}|\mathbf{W}_i;\Theta)P(\mathbf{W}_i) = P(\mathbf{u}_i|\mathbf{B})(\prod_{j=1}^{k_i} P(\mathbf{b}_j|\mathbf{w}_j^i;\Theta))P(\mathbf{W}_i) \tag{1.4}$$

The model parameters are then estimated using the EM-algorithm. Parameters related to the language model, can be initialized with use of graphone language model. Several technical issues has to be dealed with, however the Pronunciation Mixture Models can be trained on the same data as traditional ASR engines and can be used to obtain phonetic transcriptions from audio data.

Similar approach is considered in the work Reddy and Gouvêa [2011], except they do not have access to the acoustic models or phone lattices, only the word recognition mistakes. An OOV word is passed through an ASR decoder giving an n-best word recognition output. Since the words are OOVs, every hypothesis will be a recognition mistake. These mistakes are then explited, assuming that the following generative story of the recognition output for a word w holds:

1. A pronunciation baseform $\mathbf{b}$ is drawn from the distribution Θ.
2. A phonetic confusion function from the word $\mathbf{w}$ and the selected baseform $\mathbf{b}$ is applied in order to generate a phoneme sequence $\mathbf{p}$ with probability $P(\mathbf{p}|\mathbf{b},\mathbf{w})$
3. A word sequence $\mathbf{e}$ with probability $P(\mathbf{e}|\mathbf{p},\mathbf{b},\mathbf{w}) = 1$ is generated using the pronunciation lexicon.

Authors model the joint probability of hypothesis and reference word $P(\mathbf{e},\mathbf{w}) = P(\mathbf{w}\sum_b f_{e,b,w})$, where $f_{e,b,w} = P(\mathbf{e}|\mathbf{b},\mathbf{w})$ is the phonetic confusion function and it is used to estimate the distribution $P(\mathbf{b}|\mathbf{w},\mathbf{e})$. Thus they are able to derive pronunciations without access to ASR lattices, i.e. it only consideres the recognizer as a black-box.

1.3.2 Conclusion

Many approaches were introduced, that are able to convert an utterance in ortographic or audio form to its phonetic representation. $G2P$ converts the grapheme transcriptions, using only the text input. It is a well explored field of study with many different variants of realization. Although it achieves very good results nowadays, it suffers from the fact, that same groups of letters may have different pronunciations in different languages. We typically don't have access to the information which language is considered and it may be difficult to get access to sufficient number of datasets. The latter problem can be partially solved by transfer learning as proposed in **?**. Alternatively, one can derive pronunciations directly from audio signal. This approach has been also explored by some authors, however it usually requires quite low level modifications of the speech recognizer. Also, the authors used the derived pronunciations to enlarge the phonetic dictionary of the recognizer, not to improve the Text-To-Speech sytems. We explore methods of merging the mentioned approaches to combine both textual and acoustic information and its usability when confronted with human judgments.

1.4 Thesis overview

2. Title of the second chapter

2.1 Title of the first subchapter of the second chapter

2.2 Title of the second subchapter of the second chapter

Conclusion

Bibliography

Arpabet overview. `https://nlp.stanford.edu/courses/lsa352/arpabet.html`. Accessed: 2017-04-16.

Maximilian Bisani and Hermann Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451, 2008.

Michael J Dedina and Howard C Nusbaum. Pronounce: a program for pronunciation by analogy. *Computer speech & language*, 5(1):55–64, 1991.

Ronald M Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational linguistics*, 20(3):331–378, 1994.

John Lucassen and Robert Mercer. An information theoretic approach to the automatic determination of phonemic baseforms. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, volume 9, pages 304–307. IEEE, 1984.

Ian McGraw, Ibrahim Badr, and James R Glass. Learning lexicons from speech using a pronunciation mixture model. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(2):357–366, 2013.

Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311, 1997.

Mehryar Mohri. Weighted automata algorithms. In *Handbook of weighted automata*, pages 213–254. Springer, 2009.

Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data*. Citeseer, 2004.

Sravana Reddy and Evandro B Gouvêa. Learning from mistakes: Expanding pronunciation lexicons using word recognition errors. In *INTERSPEECH*, pages 533–536, 2011.

Tim Schlippe, Sebastian Ochs, and Tanja Schultz. Grapheme-to-phoneme model generation for indo-european languages. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4801–4804. IEEE, 2012.

Terrence J Sejnowski and Charles R Rosenberg. *NETtalk: A parallel network that learns to read aloud.* MIT Press, 1988.

P. Taylor. *Text-to-Speech Synthesis.* Cambridge University Press, 2009. ISBN 9781139477260. URL `https://books.google.cz/books?id=T00-NHZx7kIC`.

Paul Taylor and Amy Isard. Ssml: A speech synthesis markup language. *Speech communication*, 21(1-2):123–133, 1997.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*, 2016.

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: A fully end-to-end text-to-speech synthesis model. *arXiv preprint arXiv:1703.10135*, 2017.

Kaisheng Yao and Geoffrey Zweig. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*, 2015.

List of Figures

List of Tables

List of Abbreviations

Attachments