INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Learning Similarity Functions for Pronunciation Variations

Einat Naaman, Yossi Adi, and Joseph Keshet

Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel
jkeshet@cs.biu.ac.il

Abstract

A significant source of errors in Automatic Speech Recognition (ASR) systems is due to pronunciation variations which occur in spontaneous and conversational speech. Usually ASR systems use a finite lexicon that provides one or more pronunciations for each word. In this paper, we focus on learning a similarity function between two pronunciations. The pronunciations can be the canonical and the surface pronunciations of the same word or they can be two surface pronunciations of different words. This task generalizes problems such as lexical access (the problem of learning the mapping between words and their possible pronunciations), and defining word neighborhoods. It can also be used to dynamically increase the size of the pronunciation lexicon, or in predicting ASR errors. We propose two methods, which are based on recurrent neural networks, to learn the similarity function. The first is based on binary classification, and the second is based on learning the ranking of the pronunciations. We demonstrate the efficiency of our approach on the task of lexical access using a subset of the Switchboard conversational speech corpus. Results suggest that on this task our methods are superior to previous methods which are based on graphical Bayesian methods.

1. Introduction

Spontaneous and conversational speech are significantly different both acoustically and linguistically from read speech. One of the key differences is the vast pronunciation variations in spontaneous and conversational speech, as the speaking rate is accelerated and consequently the pronunciation becomes reduced or coarticulated. Other factors, such as the neighboring words and the speaker's style, also influence the way words are produced.

We distinguish between two types of pronunciations of a word. The typical pronunciation found in a dictionary is called *canonical* pronunciation, whereas the actual way in which speakers produce the word is called *surface* pronunciation. Spontaneous speech often includes pronunciations that differ from the one found in the dictionary. For example, pronunciations of the word "probably" in the Switchboard conversational speech corpus include [p r aa b iy], [p r aa l iy], [p r ay], etc. [1]. Fewer than half of the word productions are pronounced canonically in the phonetically transcribed portion of Switchboard [2].

In this work we propose to learn a similarity function between two pronunciation variations. The function should score the similarity between any type of pronunciations. The input can be, for example, canonical and surface pronunciations or it can be two surface pronunciations. We expect such a function to output a high number if the inputs are canonical and surface pronunciations of the same word or if the input consists of two surface pronunciations of the same word.

Such a similarity measure can be utilized in many tasks, including lexical access, word neighborhood and pronunciation scoring. In the task of lexical access, the goal is to predict which word in the dictionary was uttered given its pronunciation in terms of sub-word units [3, 4, 5]. The problem of lexical access can be handled using a pronunciation similarity function as follows: the input surface pronunciation is compared to all of the (canonical) pronunciations in the dictionary using the similarity function, and the one with the maximal similarity is predicted as the articulated word.

In the word neighborhood task the goal is to find the acoustic neighborhood density of a given word [6]. It has been suggested as an explanatory variable for quantifying errors in ASR, but is also used in linguistic studies. In most works in the psycholinguistics literature, word neighborhood is defined to be the set of words which differ by a single phone from the given word. Here we propose a different definition which is closer to the definition suggested in [6]. The word neighborhood can be defined as all the words in the dictionary in proximity to the given word, where proximity is measured using the similarity function between pronunciations.

ASR systems are based on a finite dictionary which provides each word with one or more pronunciations. The variance of pronunciations in spontaneous and conversational speech leads to a high rate of errors in ASR systems [7, 8]. The standard approach to this problem is to expand the dictionary, either by adding alternate pronunciations with probabilities, or with phonetic substitution, insertion and deletion rules derived from linguistic knowledge and learned from data. A similarity function can be used to add pronunciation variations to each word either statically or dynamically during the dictionary lookup.

Mapping between words and their possible pronunciations in terms of sub-word units was explored in light of the lexical access task [9, 4, 5]. Note that this problem is different from the grapheme-to-phoneme problem, in which pronunciations are predicted from a word's spelling, whereas in lexical access we assume a dictionary of canonical pronunciations like the one used in speech recognition. Learning word neighborhood automatically was proposed by [6].

Our work is restricted to proposing algorithms for learning the pronunciation similarity score. Specifically we propose two different deep network architectures to tackle this problem. The first is based on binary classification of two recurrent neural networks (RNNs), while the second is based on triplet networks designed to rank the pronunciations.

The paper is organized as follows. Section 2 introduces our notation and the problem definition. In Section 3 we propose two deep network architectures to learn similarity between two pronunciations. In Section 4 we present a set of experiments on the Switchboard conversational speech corpus, and in Section 5 we analyze the experimental results. We conclude the paper in Section 6.

http://dx.doi.org/10.21437/Interspeech.2017-1117

2. Problem settings

We denote a word pronunciation by a sequence of phones, $\boldsymbol{p} = (p_1, \ldots, p_N)$ where $p_n \in \mathcal{P}$ for all $1 \leq n \leq N$ and $\mathcal{P}$ is the set of all sub-word units (all phones). Naturally, N is not fixed since the number of phonemes varies across different words. We denote the set of all finite-length phone streams as $\mathcal{P}^*$.

Given two pronunciation sequences, our goal is to find a similarity function between these two sequences. Formally, our goal is to learn a function $f : \mathcal{P}^* \times \mathcal{P}^* \to \mathbb{R}$ which gets as input two pronunciation sequences and returns the similarity between the two. That is if two pronunciations $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathcal{P}^*$ are similar, then the function $f(\boldsymbol{p}_1, \boldsymbol{p}_2)$ will be high. Otherwise it will be low.

This type of function can be utilized in various tasks. For example, in the lexical access task [5] the goal is to predict which word w from a finite dictionary $\mathcal{V}$ is associated with a given surface pronunciation $\boldsymbol{p}^s$. Assume that the dictionary $\mathcal{V}$ is a list of pairs, where each pair is composed of a word w and its canonical pronunciation $\boldsymbol{p}^c$, namely $(w, \boldsymbol{p}^c) \in \mathcal{V}$. Define $sort^k$ as a function that gets a set of unordered scores and returns a vector of the top k maximal ordered scores. Similarly define $arg\ sort^k$ to be the function the returns the indices of the ordered scores. Then the best k-words can be found to be those whose canonical pronunciations get the highest similarity to the input surface pronunciation:

$$\boldsymbol{w} = \operatorname*{arg\,sort^k}_{(w,\boldsymbol{p}^c)\in\mathcal{V}} f(\boldsymbol{p}^s, \boldsymbol{p}^c), \qquad (1)$$

where $\boldsymbol{w}$ is a list of k-best words in the dictionary. Either the word with the highest unigram probability can be predicted or it can be combined to generated a prediction based on the n-gram probability.

In the word neighborhood task the goal is to find the acoustic neighborhood density of a given word [6]. This can be achieved by comparing the similarity of a canonical pronunciation of a given word to the set of all canonical pronunciations in the dictionary. Formally we propose to define word neighborhood $\mathcal{N}(w)$ for a given word w as the set of all words u in the dictionary $\mathcal{V}$ for which the similarity function is less than some threshold θ:

$$\mathcal{N}(w) = \{u \mid \forall(u, \boldsymbol{p}^u) \in \mathcal{V},\ f(\boldsymbol{p}^w, \boldsymbol{p}^u) < \theta\}, \qquad (2)$$

where $\boldsymbol{p}^w, \boldsymbol{p}^u \in \mathcal{P}^*$ are the canonical pronunciations of the word w and u respectively.

3. Network architecture

Is this section we suggest two network architectures to learn the similarity function. Both architectures are based on Recurrent Neural Networks (RNNs) [10]. The first one is based on Siamese RNNs which were designed as a binary classifier and were trained to minimize the negative log likelihood loss function. The second architecture is based on three identical RNNs, which are combined together to create a ranking architecture and optimized with a ranking loss function. We also tried several sequence-to-sequence architectures [11, 12], but since all of them led to poor results in terms of Word Error Rate (WER), we will not discuss them in the paper.

3.1. Binary loss function

The first architecture is a network that is designed to learn a mapping between two pronunciations using a binary classifier, which is trained to predict whether two pronunciations are of the same word. Each example in the training set of m examples, $S = \{(\boldsymbol{p}_i^s, \boldsymbol{p}_i^c, y_i)\}_{i=1}^m$, is composed of a surface pronunciation $\boldsymbol{p}^s \in \mathcal{P}^*$, a canonical pronunciation $\boldsymbol{p}^c \in \mathcal{P}^*$ and a binary label $y \in \{-1, +1\}$, which indicates if both pronunciations are of the same word or not.

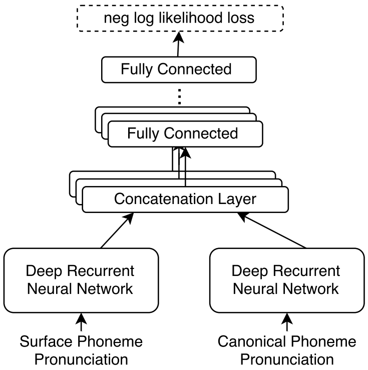
neg log likelihood loss
Fully Connected
Fully Connected
Concatenation Layer
Deep Recurrent Neural Network
Deep Recurrent Neural Network
Surface Phoneme Pronunciation
Canonical Phoneme Pronunciation

Figure 1: *Network architecture of the binary loss network. The two RNNs share the same parameters.*

We would like to train a neural network to learn this binary mapping. In order to do so, we encode both pronunciations using two RNNs with a shared set of parameters. The input to each of the RNNs is a sequence of phones, which represents either surface or canonical pronunciations, and the output is a real vector. Denote by $\boldsymbol{v}^c$ and $\boldsymbol{v}^s$ the output of the RNNs for the canonical and the surface representations, respectively. Then, both vectors are concatenated $[\boldsymbol{v}^c, \boldsymbol{v}^s]$ and are fed to three fully connected layers with shared parameters over time. The output is a series of vectors, one for each time step. These are all concatenated[1] and are fed to a fully-connected layer followed by a softmax layer. The similarity function f is the output of the softmax layer, and it can be interpreted as a probability function. The whole network is trained so as to minimize the negative log likelihood loss function. The network architecture is depicted schematically in Figure 1.

3.2. Ranking loss function

A different approach to learning the similarity function is to score the similarity between a given pronunciation and other pronunciations according to their ranking. In order to do so we use a slightly different training set than the one used to train the binary network. Each example in the new training set is composed of a triplet: a surface pronunciation $\boldsymbol{p}^s \in \mathcal{P}^*$, a positive canonical pronunciation $\boldsymbol{p}^+ \in \mathcal{P}^*$, and a negative canonical pronunciation $\boldsymbol{p}^- \in \mathcal{P}^*$. The positive canonical pronunciation is the canonical pronunciation associated with the surface pronunciation $\boldsymbol{p}^s$, and the negative canonical pronunciation is a canonical pronunciation of a different word. Overall the training set of m examples is denoted $S = \{(\boldsymbol{p}_i^s, \boldsymbol{p}_i^+, \boldsymbol{p}_i^-)\}_{i=1}^m$. As in the previous model, we represent each of the three pronunciations using an RNN. The output of the RNN is fed into two fully-connected layers and is considered to be the *pronunciation embedding*. Pronunciation embedding is a function g that maps a pronunciation $\boldsymbol{p} \in \mathcal{P}^*$ to a fixed size vector, $\boldsymbol{u} \in \mathbb{R}^n$, where $\boldsymbol{u} = g(\boldsymbol{p})$. We measure the closeness between the two embeddings, $\boldsymbol{u} \in \mathbb{R}^n$ and $\boldsymbol{v} \in \mathbb{R}^n$, using the cosine distance

[1] We apply sequence padding when needed.

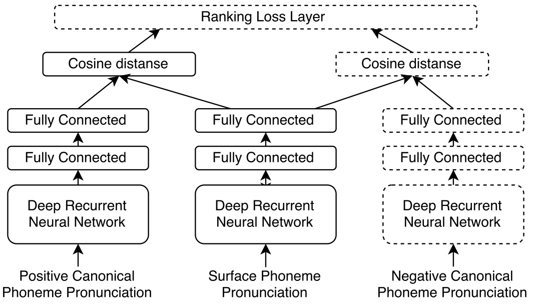
Ranking Loss Layer
Cosine distanse
Cosine distanse
Fully Connected
Fully Connected
Fully Connected
Fully Connected
Fully Connected
Fully Connected
Deep Recurrent Neural Network
Deep Recurrent Neural Network
Deep Recurrent Neural Network
Positive Canonical Phoneme Pronunciation
Surface Phoneme Pronunciation
Negative Canonical Phoneme Pronunciation

Figure 2: *Network architecture of the ranking model. The three RNNs have share parameters and output an embedding vector.*

score [13]:

$$d_{\cos}(\boldsymbol{u},\boldsymbol{v}) = \frac{1}{2}\left(1 - \frac{\boldsymbol{u}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}\right).$$

The cosine distance score between two embeddings is close to 0 if the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are close, and close to 1 if they are far apart. Formally, the similarity function between two pronunciations $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is defined as 1 minus the cosine distance

$$f(\boldsymbol{p}_1,\boldsymbol{p}_2) = 1 - d_{\cos}(g(\boldsymbol{p}_1), g(\boldsymbol{p}_2)). \qquad (3)$$

During training we minimize the hinge loss over the cosine distance so that the score of the related canonical-surface pronunciations is higher than the score of the unrelated canonical-surface pronunciation by a margin of at least γ, where $\gamma \in \mathbb{R}_+$ is a positive scalar. Formally, the loss function is defined as [14, 13]:

$$\ell(\boldsymbol{p}^s,\boldsymbol{p}^+,\boldsymbol{p}^-) = \max\{0, \gamma - f(\boldsymbol{p}^s,\boldsymbol{p}^+) + f(\boldsymbol{p}^s,\boldsymbol{p}^-)\}, \quad (4)$$

where f is the similarity function defined in Eq. (3), and γ is the margin parameter. The network architecture is depicted schematically in Figure 2.

This architecture has three advantages over the binary architecture: (i) the ranking optimization criterion is closer to the notion of similarity function: related pronunciations get higher score than unrelated ones; (ii) as a by-product we introduce the *pronunciation embedding*, which maps a sequence of phones to a fixed size vector; and (iii) we use many negative surface pronunciations for each positive surface pronunciation and increase our training data. We found that it has a great impact on the model's performance, and we analyze it in Section 5.

4. Experiments

We evaluated the proposed architectures on the lexical access task, where we would like to predict the word in the dictionary that is associated with a given surface pronunciation. All experiments are conducted on a subset of Switchboard conversational speech corpus that has been labeled at a fine phonetic level [1]; these phonetic transcriptions are the input to our similarity models. The data subsets, phone set $\mathcal{P}$, and dictionary $\mathcal{V}$ are the same as those previously used in [9, 4, 5]. The dictionary contains 5,117 words, consisting of the most frequent words in Switchboard. The base-form uses a similar, slightly smaller phone set (lacking, e.g., nasalization). We used the same partition of the corpus as in [9, 4, 5] into 2,942 words in the training set, 165 words in the development set, and 236 words in the test set.

Results are presented in terms of the word error rate when the top k predictions are considered. This is denoted by WER@k. Table 1 summarizes the results for all our architectures. For each architecture we tested three types of RNNs: LSTM with one layer, LSTM with two layers (2-LSTM), and bidirectional LSTM with two layers (BI-2-LSTM). We optimize all our models using Adagrad [15] with learning rate value of 0.01. We use ReLU [16] as an activation function after each fully connected layer. For the ranking models we use a margin value of $\gamma = 0.3$. All hyper-parameters were tuned on a validation set.

Table 1: *WER for the binary-loss model and the ranking-loss model on the lexical access problem.*

	Models	**Test**	
		WER@1	**WER@2**
	dictionary lookup [2]	59.3%	-
	dictionary + Levenshtein dist.	41.8%	-
	Jyothi et al., 2011 [4]	29.1%	-
	Hao et al., 2012 [5]	15.2%	-
Binary	LSTM	20.8%	18.2%
	2-LSTM	24.6%	22.5%
	BI-LSTM	21.6%	19.5%
Rank	LSTM	25.9%	16.5%
	2-LSTM	22.9%	14.8%
	BI-LSTM	23.3%	15.3%

For comparison we added to Table 1 the word error rate of other algorithms for lexical access: a dictionary lookup with and without Levenshtein distance [9], a dynamic Bayesian network (Jyothi *et al.*, 2011 [4]), and discriminative structured prediction model (Hao *et al.*, 2012 [5]). It can be seen from the table that both of our models outperform the dictionary lookup approaches and the model which is based on dynamic Bayesian networks [4]. However the discriminative structured prediction model [5] performs much better than any of our models. The discriminative model was trained specifically for the lexical access task with a unique set of feature maps, but it cannot be used as a similarity score between two pronunciations.

The performance of the binary loss model and the ranking model are almost the same, with the binary loss performing slightly better than the rank loss. The reason is likely that it was trained to maximize the probability that two pronunciations are related and not to match a similarity score.

5. Analysis

In this section we analyze the performance of the ranking model. We investigate the effect of the number of negative examples, the effect of the embedding size and present some of the model's outputs and the way it errs.

5.1. The effect of the number of negative samples

Recall that the ranking loss model was trained on triplet made of a canonical pronunciation of a word, a surface pronunciation of the word (positive sample), and a surface pronunciation which is not associated with the word (negative sample). In our experiments the negative surface pronunciation was a surface pronunciation of a random word.

Deep neural networks require a lot of training data in order to converge to a good local minimum. We expanded the training set by using many different negative samples for each positive samples. In order to examine if this approach leads to a better performance, we trained the network several times with

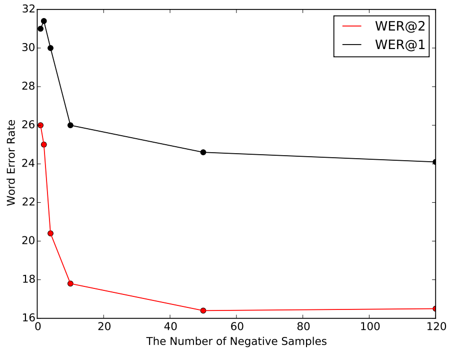

Figure 3: *WER@1 and WER@2 of the test set as a function of the number of samples per example.*

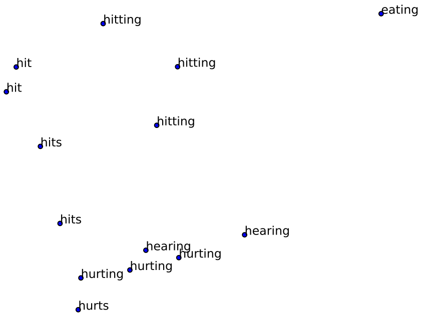

Figure 4: *Two-dimensional t-SNE projection of the representation vectors (zoom to a specific area.)*

a different number of negative samples per positive sample and evaluated the performance on the test set.

Figure 3 shows WER@1 and WER@2 of the lexical access task, as a function of the number of negative samples per positive one. Notice that when adding more examples the error rate keeps decreasing until the limit of 50 negative samples, from this point the error rate stays roughly the same.

5.2. The effect of the embedding size

Next, we investigated the effect of the pronunciation embedding size n, i.e., the size of the output of the last layer after the RNN. We tried different embeddings sizes, and evaluated their performance on the validation and test sets. Table 2 summarizes the results. From the table we see that embedding of size 40 is too small, but the performance of embeddings of size 80 and above are all good. We found the embedding of size 120 to yield the best performance.

Table 2: *Performance (WER) for different embedding sizes.*

Dim.	**Test**		**Validation**	
	WER@1	**WER@2**	**WER@1**	**WER@2**
150	24.2%	16.9%	18.8%	12.7%
120	25.9%	16.5%	18.2%	11.5%
80	25.0%	17.4%	21.2%	14.6%
40	27.1%	18.7%	24.9%	15.8%

5.3. Visualization

Lastly, we performed a few visualizations in order to get a sense of what the model learned. In Figure 4 we visualized a subset from the embedding space of the canonical pronunciations in the dictionary, using t-SNE [17] for dimensionality reduction. The words which have a similar pronunciation appear to be close in the embeddings space.

In Table 3 we present the 4 most similar words according to the learned similarity function for the words *sense, write, die, male*, and *their*. Again, we can see that the most similar words in the embedding space are the words which contain a similar phone sequence.

Table 3: *Four most similar words from the dictionary computed in the embedding space.*

word	**neighborhood**
sense	cents, since, sent, sentence
write	right, ride, righty, writing
die	diet, died, dying, idea
male	mail, meal, may, makes
their	there, they're, there'd, there're

In Table 4 we illustrate the predictions of the model for the lexical access task for hard cases of surface pronunciations. The table shows the first three predictions of the model, ordered (left to right) from most similar to least similar. The table is separated into two panels: the upper panel shows the correct predictions made by the model and the lower panel shows incorrect predictions. It can be seen that both the correct and the incorrect prediction are hard to classify, even for a human. In the lower panel, it seems that there might be an error in the transcription (e.g., $[s, tcl, t, ao, r]$ is more similar to the predicted word *store* rather than to labeled word *start*), or that we have reached the limit of the possible discrimination in this task (e.g., $[w_n, ah_n, n]$ can equally be predicted as either *want* or *won*.

Table 4: *Prediction of the model for hard cases of surface pronunciations.*

surface pronunciation	**desired word**	**predicted words**
$[bcl, b, ao]$	bought	bought, bob, ball
$[m, ey1, ey2, dcl, jh, er]$	major	major, mayor, made
$[f, ay1, ay2, n, ih_n, ng]$	finding	finding, fighting, flying
$[pcl, p, r, aa, er]$	proper	proper, prior, property
$[n, pcl, p, eh, er, z]$	peppers	peppers, persons, present
$[s, tcl, t, ao, r]$	start	store, star, sent
$[eh, r, uw, ay1, ay2]$	everybody	iraq, dry, era
$[w_n, ah_n, n]$	want	won, one, want
$[pcl, p, uw, el]$	people	pool, pull, people

6. Conclusion

We presented two very different architectures to learn similarity between two phone streams. The first architecture learns the alignment between phone streams that represent close pronunciations. The second architecture is designed to learn a mapping of a phone stream to a vector space, such that close pronunciations will have close representation in the output vector space.

Future work will explore similarity between other sub-word units. Specifically we would like to propose a similarity function between articulatory features, and analyze it in the light of articulatory phonology [18].

7. Acknowledgment

We would like to thank Karen Livescu, Preethi Jyothi and the anonymous reviewers for their helpful comments.

8. References

[1] S. Greenberg, J. Hollenback, and D. Ellis, "Insights into spoken language gleaned from phonetic transcription of the Switchboard corpus," in *Proceeding of the 4th International Conference on Spoken Language Processing (ICSLP)*, 1996.

[2] K. Livescu, "Feature-based pronunciation modeling for automatic speech recognition," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[3] L. Fissore, P. Laface, G. Micca, and R. Pieraccini, "Lexical access to large vocabularies for speech recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 8, pp. 1197–1213, 1989.

[4] P. Jyothi, K. Livescu, and E. Fosler-Lussier, "Lexical access experiments with context-dependent articulatory feature-based models," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011.

[5] H. Tang, J. Keshet, and K. Livescu, "Discriminative pronunciation modeling: A large-margin, feature-rich approach," in *The 50th Annual Meeting of the Association of Computational Linguistics (ACL)*, 2012.

[6] P. Jyothi and K. Livescu, "Revisiting word neighborhoods for speech recognition," in *ACL MORPHFSM Workshop*, 2014.

[7] D. Jurafsky, W. Ward, Z. Jianping, K. Herold, Y. Xiuyang, and Z. Sen, "What kind of pronunciation variation is hard for triphones to model?" in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001.

[8] M. Saraçlar and S. Khudanpur, "Pronunciation change in conversational speech and its implications for automatic speech recognition," *Computer Speech and Language*, vol. 18, no. 4, 2004.

[9] K. Livescu and J. R. Glass, "Feature-based pronunciation modeling with trainable asynchrony probabilities." in *Proceeding of the 8th International Conference on Spoken Language Processing (ICSLP)*, 2004.

[10] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[11] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[12] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," *arXiv preprint arXiv:1506.00196*, 2015.

[13] H. Kamper, W. Wang, and K. Livescu, "Deep convolutional acoustic word embeddings using word-pair side information," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4950–4954.

[14] J. Keshet, D. Grangier, and S. Bengio, "Discriminative keyword spotting," *Speech Communication*, vol. 51, no. 4, pp. 317–329, 2009.

[15] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[16] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML)*, 2010, pp. 807–814.

[17] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[18] C. P. Browman and L. Goldstein, "Articulatory phonology: An overview," *Phonetica*, vol. 49, no. 3-4, pp. 155–180, 1992.