Learning Personalized Pronunciations for Contact Name Recognition

Antoine Bruguier, Fuchun Peng, Françoise Beaufays

Google Inc., USA

{tonybruguier,fuchunpeng,fsb}@google.com

Abstract

Automatic speech recognition that involves people's names is difficult because names follow a long-tail distribution and they have no commonly accepted spelling or pronunciation. This poses significant challenges to contact dialing by voice. We propose using personalized pronunciation learning: people can use their own pronunciations for their contact names. We achieve this by implicitly learning from users' corrections and within minutes making that pronunciation available for the next voice dialing. We show that personalized pronunciations significantly reduce word error for difficult contact names by 15% relatively.

Index Terms: speech recognition, pronunciation, personalization

1. Introduction

Voice contact dialing and texting has become increasingly important due to the popularity of smartphones [1]. Compared to other speech recognition tasks such as voice search and dictation, voice contact dialing poses some unique challenges. The key part of voice contact dialing is to get the contact name correctly recognized, otherwise the call cannot be initialized or is initialized to a wrong person. This is very challenging for several reasons. First, name utterances are short; they usually contain very few words, for example, "call Jennifer". This means that there is not much context that a language model can leverage. Second, contact names usually have many variants and many names have foreign origins. Thus it is hard for other people without this knowledge to pronounce them correctly. For example, the Chinese name "Fuchun" is often pronounced as "Fortune" by people who don't speak Chinese natively. Thus, recognition accuracy of contact names usually is far worse than other voice search queries [2, 3].

To alleviate these problems, Aleksic et al. [2] use contact biasing toward the names on the contact list. This can be considered language modeling personalization. They then look up each name in a pronunciation dictionary and, in case the name is not present, use a pronunciation prediction model [4]. If the pronunciation for a name is wrong, the recognizer will systematically misrecognize that name, resulting in a very frustrating user experience.

It is prohibitively costly for humans to transcribe the pronunciations manually for all the words in an ASR system and it is much easier to collect written text. For American English, we have a human pronunciation for about of 12% the words.

For the words not in our dictionary, we use a machine learning grapheme to phoneme (G2P) model to generate pronunciations [4]. The G2P has a word accuracy of only 65%. The low number can be explained by the variety of the word we are predicting (common names, personal names, acronyms, street names, etc...) and that we also predict the English pronunciation of words of foreign origin. When a pronunciation is incorrect, it is more difficult for the word to be recognized because at least one phoneme doesn't match what the users are saying. Thus, even if a word is in vocabulary, it can be difficult for our system to recognize it.

For names, the problem is even more acute for two reasons. First they follow a long-tail distribution, meaning that they are more likely to be missing from our pronunciation dictionary. Second, even if they are present in our dictionary, a user may not pronounce the name according to the generally accepted way.

Data driven approaches have successfully learned pronunciations from data [5, 6, 7] improving accuracy to 90%. Previous efforts in pronunciation learning focused on learning pronunciation offline from a large amount of data, and then applied the learned pronunciations to all users. This aggregated approach is effective yet the benefit of data driven pronunciation learning is not fully maximized. It requires enough samples to generalize the learned pronunciations to new utterances. Thus, we have to throw away a lot of pronunciations that are reasonable for certain users, but not common enough to be used by all users.

This is especially true for contact names since contact names have significant variations in pronunciations. Different people may have different pronunciations for the same names. For example, for "Johan Schalkwyk" the correct pronunciation is "j U h A n" for "Johan" and "S O l k w @ k" for "Schalkwyk" [1], but people can also say "dZ oU h A n" for "Johan" and "s k A k w @ k" for "Schalkwyk". For speech recognition, even though "dZ oU h A n s k A k w @ k" is not the canonical pronunciation, we still need to make recognition work.

In order to estimate how severe the lack of dictionary pronunciation is, we collected all the learned pronunciations from the logs in the form of a (word, pronunciation) tuple. We restricted the sampling to examples where the contact name wasn't recognized properly. We then counted the number of users where this tuple appears. If a tuple appears multiple times for a given user, it counts as one. We found 98% of contact pronunciations only occur fewer than 10 users (Table 1). These pronunciations appear in so few distinct users that they are probably not commonly accepted. If the reason that we don't recognize the name properly is that it has a commonly accepted pronunciation but we don't know what it is, the histogram would have been heavier in the bin with a larger number of distinct pair. If that had been the case, it would have made sense to merge these canoncial pronunciations across users.

To address this long-tail problem, we propose using personalized pronunciations. In particular, we propose a method to identify the difficult names where ASR originally failed and the user manually provided the right contact name, we learn the pronunciations for such names on the fly and apply them to future recognition of this user's utterances shortly after they are

[1] We use x-sampa phone notation, https://en.wikipedia.org/wiki/X-SAMPA

Number of distinct pair	Fraction of traffic
1	81.10%
2	8.68%
3	3.39%
4	1.74%
5	1.01%
6	0.71%
7	0.53%
8	0.35%
9	0.35%
10 or more	2.15%

Table 1: Distribution of pair of (word, pronunciation).

learned. We propose a practical architecture and algorithm improvements, and demonstrate significant improvements in real data. To our knowledge, we are the first to propose using personalized pronunciations for ASR and demonstrate its effectiveness over state of the art recognition engines.

2. Related Work

Our work falls into general pronunciation modeling such as grapheme to phoneme modeling (G2P) [4, 8, 9]. Different from these grapheme to phoneme prediction based on pure text, our work learns pronunciations from both audio and text.

Our work is also closely related to pronuciation learning from audio [5, 10]. The difference is that here we learn the pronunciations for personalization. Personalization poses new challenges in both architecture and learning algorithms. We have to implement a new real time error detection mechanism and an architecture that makes use of personalized pronunciations. We also need to make changes in learning algorithms to handle varieties of names.

Another line of related work is personalized name recognition based on contact biasing [2, 11] and salient n-gram biasing [12]. Those can be considered as language model personalization. Our work is built on top of that, but focuses on prounciation personalization.

Our work is also related to acoustic model adaptation [13]. Acoustic model adapatation is difficult as it requires enough personalized data to adapt. Instead, we adapt at the pronunciation level, which does not need to retrain an acoustic model and can be done in real time and from a single example.

3. Learning Personalized Pronunciation

3.1. Audio Driven Pronunciation Learning

The audio driven pronunciation learning algorithm is the backbone for our system. We extend the approach of [5].

In speech recognition, we wish to find the word sequence W^* that maximizes the likelihood of the acoustic observations, $\mathcal{X}$

$$W^* = \arg\max_i P(W_i|\mathcal{X}) \tag{1}$$

$$= \arg\max_i P(\mathcal{X}|W_i)P(W_i). \tag{2}$$

Assuming that different phoneme sequences S_j^i can underlie the same word sequence W_i (pronunciation dictionaries often allow multiple pronunciations per word), and with the Viterbi approximation, Eq. 2 becomes

$$W^* = \arg\max_{i,j} P(\mathcal{X}, S_j^i|W_i)P(S_j^i) \tag{3}$$

$$= \arg\max_{i,j} P(\mathcal{X}|S_j^i)P(S_j^i|W_i)P(W_i). \tag{4}$$

For the purpose of pronunciation learning, we instead assume that the word sequence W_i corresponding to the acoustic sequence $\mathcal{X}$ is given, but multiple pronunciations are available. We wish to find the pronunciation sequence S^* that maximizes the likelihood of the acoustic data, under the assumption of the given word sequence $\hat{W}$:

$$S^* = \arg\max_j P(\mathcal{X}|S_j)P(S_j|\hat{W})P(\hat{W}) \tag{5}$$

$$= \arg\max_j P(\mathcal{X}|S_j). \tag{6}$$

where $P(S_j|\hat{W})$ and $P(\hat{W})$ can be dropped if we assume equal priors on pronunciations and words, respectively. Essentially, we force-align the transcript against the audio to extract the best phoneme sequence as pronunciation.

3.2. Architecture for Personalized Pronunciation Learning

Personalized pronunciation learning consists of the following two parts: *learning* and *serving*, as illustrated in Figure 1.

The entire pronunciation learning process has a latency in the order of minutes, i.e., when the original recognition failed and the user made a correction, in a few minutes the learned pronunciation will go into effect. Our system listens to correction events using a publisher-subscriber approach [14]. Once the events are received, we join with the audio data and learn the personalized pronunciation. The vast majority of the latency is due to the globally replicated nature of the storage and waiting for a consistent state. The active pronunciation learning code runs for a time on the order of seconds.

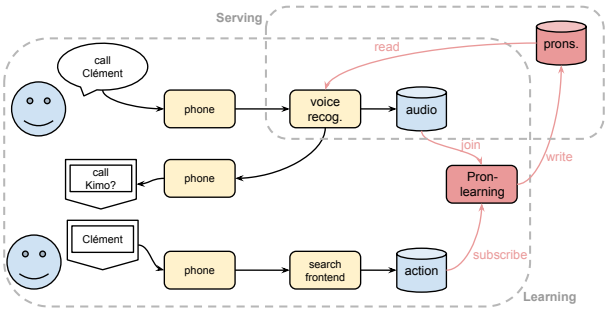

Figure 1: Architecture of personalized pronunciation of *learning* and *serving*.

The *learning* part consists of these components:

1. Extract audio and its corresponding contact names.
2. Construct pronunciation learning request.
3. Store the learned pronunciations for the contacts in the person's own database.

In *serving*, recognizer extracts the personalized pronunciation from the user's account to construct decoder graph. We will describe each of these components in more details below.

3.2.1. Extract audio and its corresponding contact names

When the correct contact name is not recognized, a user interface provides the option to select the intended contact from the contact list. For example, a user uttered the command "Call Clément" and the ASR system misrecognized the name and transcribed the command as "Call Kimo". Since "Kimo" is not in the contact list, there is a user interface asking the user to select a contact to make the call. The user manually corrects with a tap and selects the correct contact ("Clément") and the phone call is initiated. If the contact "Kimo" had been in the contact list, the user would have had the option to interrupt the dialing and make a manual correction.

The actions of the users were previously not used beyond completing the call and a correction did not result in a changed behavior of the recognizer. To make use of this information, we use a PubSub service [14] to monitor these actions. When the subscriber identifies these sequence of actions, we can associate the corrected contact names with the original audio.

3.2.2. Construct Pronounciation Learning Request

A pronunciation learning request consists of audio and its corresponding transcript. The transcript will be aligned with audio. While we do know the contact's name as it appears in the list, we do not know whether the user refered to that contact by her first name, last name, or a combination of the two parts. This complicates the pronunciation learning algorithm as we need to provide the transcript truth [5], but the exact transcript is unknown.

To solve this issue, we extended the algorithm of [5] to accept multiple transcripts. We create multiple paths for each possibility as shown in figure 2. In this example, the transcript is "call $NAME" and the user's action suggests that she intended to call "Antoine Bruguier". However, we do not know what the user said for the name and thus we hypothesized that it could be either "Antoine", "Bruguier", "Antoine Bruguier", or "Bruguier Antoine". We inserted these four possible branches in the otherwise linear FST to obtain the FST shown here. During the decoding, we keep track of which path has the lowest acoustic cost, and the name parts that belong to this path are the ones for which pronunciations are learned.

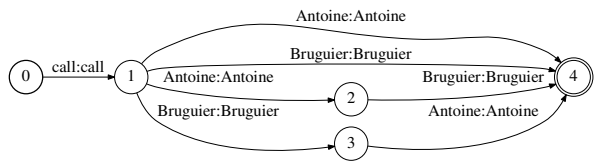

Figure 2: Example of construction of the G (grammar) FST for pronunciation learning.

Another improvement to the learning algorithm is that we limit pronunciation learning to only the contact names, and for the other words in the transcript (such as call, text, etc), we use default pronunciations. This is done by not generating candidates for words we do not want to learn and instead using dictionary pronunciations. This approach restricts the number of allowed paths in the force alignment step and acts as a regularization.

For languages other than American English, the construction of the FST may be more complicated. For example, some languages such as Russian use declensions and we need to take into account the correct case of the words. Chinese names are not separated by spaces. For the rest of this paper, we will focus on American English recognition.

3.2.3. Personalized Pronunciation Storage

The learned pronunciations can be stored on the server associated with the user's account, or it can be stored locally on device. The server storage is globally replicated for fast access.

3.2.4. Replacing Decoder Graph

At *serving* time, we retrieve the pronunciation from users personal account, and construct a lexicon-grammar (LG) FST for the contact names [15]. The G graph contains all the names in the user contact list and is constructed in the same way as in [3]. The lexicon (L) FST contains the pronunciations for the words in Grammar (G) FST. We use personalized pronunciation if they are available. If no personalized pronunciation is available, we revert to the previous behavior of [3] and use a dictionary or G2P pronunciation. We then compose the L and G graphs to get LG. The entire process happens on the order of miliseconds, to allow for real-time recognition.

3.3. User experience

From the users' prospective, the interaction is seamless. When a contact recognition fails, they can manually select the correct contact and continue with their intended action. The learning code is triggered without any further interaction and the pronunciation is used next time they issue a voice query.

4. Experiments

4.1. Privacy Consideration

For each utterance, we only recorded the name of the intended contact and not the full contact list. We also anonymized the utterances, making it impossible to track a user.

4.2. Datasets and Metrics

We built two data sets by randomly sampling utterances. We stripped all the personal information for the user, and thus we only had access to the audio and the person the user intended to call.

The first data set contained utterances sampled from user corrections. Since we use contact list for language model biasing [2] yet we do not access the user's contact list, we have to simulate a contact list. To do so, we follow the same strategy used in [2, 3] by grouping utterances together. For each utterance we hashed its information into a bucket number. We then collected all the utterances in that bucket and create a contact list from all the person contacted. For example, if we had 4,000 utterances, we assigned to each of them a bucket number between 1 and 20, resulting in 20 groups of about 200 utterances each. Following previous practice [3], we chose 200 as the number of contacts for experiments. In each of the groups, we collected all the persons called, resulting in a contact list of about 200 for each of the utterance. Thus, in each utterance there was the intended contact present, and about 199 other contacts that the user did not intend to reach. These groups, coming from randomly-sampled utterances were treated as containing utterances coming from a single-user, despite this not being the case.

The second data set contains only names, for example, "Françoise Beaufays". This data set is not for voice dialing. Instead, the data was mined from voice search logs for evaluating

contact biasing purpose [3]. We use it here to evaluate personalized pronunciation, to see if the improvements on names can transfer to voice search queries.

4.3. Results

Our first experiment was on sampled traffic data that was personalized (table 2). We measured performance in terms of word error-rate (WER) on 3,800 utterances, with about 200 contacts per utterance. The utterances had an action (e.g. "call ...", "send a text to ...") and a name. Adding personalized pronunciations reduces the WER by 3% over simple contact biasing, a 15% relative reduction. For every 6 utterances where the recognition was improved, 1 utterance was made worse. Similar improvements (table 3) are observed a data set that consisted of only a name (without any action words).

Experiment	WER
Contact biasing, dictionary pronunciations	22.6
Contact biasing, personalized pronunciations	19.6

Table 2: Word error rates (WER) on a data set of actions that contained a name, with and without personalized pronunciations

.

Experiment	WER
Contact biasing, dictionary pronunciations	23.3
Contact biasing, personalized pronunciations	20.3

Table 3: Word error rates (WER) on a data set with only name, with and without personalized pronunciations

.

5. Discussions

5.1. Error Analysis

A win/loss ratio of 6 to 1 indicates that personalized pronunciation overall is a great success, but it is not perfect. Not surprisingly, most names with improved recognition are foreign names where the G2P pronunciations were clearly wrong. The personalized pronunciations learned from audio are much better. Below are two illustrative examples (we truncated the names to preserve our users' privacy).

Case 1: "Call Ghaythan" was recognized as "Call Nathan", because the pronunciation for "Ghaythan" was `dZ i eI tS eI w aI T @ n`. This pronunciation does not match what the users said, and the acoustic score is very low. We learned the personalized pronunciation from audio as `g eI D @ n`, which reflects what the user said.

Case 2: "Call Maa" was recognized as "Call my". The G2P predicted `E m eI eI` for "Maa" (spelling out the letters) and we had the entry `m aI` for "my". Since the user said `m @`, the (incorrect) transcription "Call my" was used, because `m @` is closer to `m aI` than `E m eI eI`. The issue was fixed after learning the correct pronunciation.

When looking at the failure cases, some are actually judgement calls. For example, *Call Dan* becomes *Call Daniel* when the audio says "`d { n`". This is marked as regression because "Dan" reflects the audio. However, the entry in the contact list is actually "Daniel". We learned pronunciation "`d { n`" for "Daniel". Thus after personalized pronunciation when people say call "call Dan", it becomes "call Daniel" and will trigger call dialing correctly.

A real regression case happens when the transcript contains foreign words that can be pronounced close to other words in English. For example, *Call La Casa* becomes *Call Mi Casa*. This utterance mixes English and Spanish, but our systems cannot switch language mid-utterance. In this case, English was used throughout for pronunciation learning. The forced alignment used an English acoustic model, phoneme set, and candidate generator. The result is that we learn the pronunciation for *Mi* to be `m @`. Language mixing is currently an open issue.

5.2. Impact of Multiple Transcripts

We wanted to evaluate the impact of multiple-transcript when learning personalized pronunciations. We ran two experiments. In the first experiment, we did insert the multiple transcripts while in the second one, we forced the transcript to be first name followed by last name (despite the fact that this didnt always correspond to what was said). We saw the WER slightly regressed from 19.6 to 19.8. Although the regression is not huge, it shows the necessity of the multiple transcripts.

5.3. Issues Beyond Pronunciations

Personalized pronunciation reduces WER from 22.6 to 19.6, which is substantial. There are still issues with names. In some cases, we close the microphone too early and the name is cut out. In other cases, even though we learn the correct pronunciation for the contact name, a misrecognition still occurs. Some users write their contact name in other scripts than the one for the language they speak. For example, they write "Samuel" with Hebrew script and use voice commands in English. This means that perfect contact name recongition requires work beyond pronunciations.

6. Conclusions

We have proposed learning personalized pronunciation for contact names. We presented an architecture based on Pub-Sub [14] to extract users' corrections for contact recognition. The learned pronunciation goes into effect in minutes. Experiments on real contact dialing data show 15% WER reduction. This is just the first piece of work we have done for pronunciation personalization. The framework can be easily extended to other scenarios beyond calling contacts, such as business names and location names.

7. References

[1] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Strope, "Google Search by Voice: A case study," *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*, 2010.

[2] P. Aleksic, M. Ghodsi, A. Michaely, C. Allauzen, K. Hall, B. Roark, D. Rybach, and P. Moreno, "Bringing contextual information to google speech recognition," in *Interspeech, International Speech Communications Association*, 2015.

[3] P. Aleksic, C. Allauzen, D. Elson, A. Kracun, D. M. Casado, and P. J. Moreno, "Improved recognition of contact names in voice commands," in *Proceedings of ICASSP*, 2015.

[4] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *Proceedings of ICASSP*, 2015.

[5] A. Rutherford, F. Peng, and F. Beaufays, "Pronunciation learning for named-entities through crowd-sourcing," in *Proceedings of Interspeech*, 2014.

[6] K. Rao, F. Peng, and F. Beaufays, "Automatic pronunciation verification for speech recognition," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[7] Z. Kou, D. Stanton, F. Peng, F. Beaufays, and T. Strohman, "Fix it where it fails: Pronunciation learning by mining error corrections from speech logs," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[8] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communications*, vol. 50, no. 5, pp. 434–451, 2008.

[9] X. Li, A. Gunawardana, and A. Acero, "Adapting grapheme-to-phoneme conversion for name recognition," in *Proceedings of ASRU*, 2007.

[10] L. Lu, A. Ghoshal, and S. Renals, "Acoustic data-driven pronunciation lexicon for large vocabulary speech recognition," in *In proceedings of ASRU*, 2013.

[11] I. McGraw, R. Prabhavalkar, R. Alvarez, M. G. Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays, and C. Parada, "Personalized speech recognition on mobile devices," in *Proceedings of ICASSP*, 2015.

[12] K. Hall, E. Cho, C. Allauzen, F. Beaufays, N. Coccaro, K. Nakajima, M. Riley, B. Roark, D. Rybach, and L. Zhang, "Composition-based on-the-fly rescoring for salient n-gram biasing," in *INTERSPEECH*, 2015.

[13] Z. Wang, T. Schultz, and A. Waibel, "Comparison of acoustic model adaptation techniques on non-native speech," in *Proceedings of ICASSP*, 2003.

[14] Google, "Google cloud pub/sub," in *https://cloud.google.com/pubsub/overview*, 2016.

[15] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," in *Computer Speech and Language*, 2002.