A Computational Model for Phonetically Responsive Spoken Dialogue Systems

Eran Raveh[1,2], Ingmar Steiner[1–3], Bernd Möbius[2]

[1]Multimodal Computing and Interaction, Saarland University, Germany
[2]Computational Linguistics & Phonetics, Saarland University, Germany
[3]German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany
lastname@coli.uni-saarland.de

Abstract

This paper introduces a model for segment-level phonetic responsiveness. It is based on behavior observed in human-human interaction, and is designed to be integrated into spoken dialogue systems to capture potential phonetic variation and simulate convergence capabilities. Each step in the process is responsible for an aspect of the interaction, including monitoring the input speech and appropriately analyzing it. Various parameters can be tuned to configure the speech handling and adjust the response style. Evaluation was performed by simulating simple end-to-end dialogue scenarios, including analyzing the synthesized output of the model. The results show promising ground for further extensions.

Index Terms: phonetic convergence, responsive spoken dialogue systems, human-computer interaction

1. Introduction

Phonetic convergence is defined as an increase in segmental and suprasegmental similarities between two speakers [1]. It has been observed in conversational human-human interaction (HHI) scenarios on the segmental level [2, 3]. Investigating phonetic convergence in human-computer interaction (HCI) paradigms is fundamentally different than in HHI, since the computer needs to be aware of potential changes in speech over time, as well as being able to produce them. While in HHI any interlocutor can initiate such changes, in HCI the artificial interlocutor is generally expected to be attentive to the human interlocutors' behavior (and potentially even to distinguish between different human interlocutors). To this end, the interactive system must be able to define speech characteristics prone to changes, detect and trace them on the fly, and finally, dynamically produce speech in which these changes are realized.

In addition to the differentiation between HHI and HCI paradigms, it is important to distinguish between *imitation* and *convergence*. An imitating system would simply try to reproduce the human interlocutor's speech characteristics every time it generates speech, while ignoring previous utterances, abrupt changes, etc. However, a system with convergence capabilities would gradually accumulate speech evidences and match its output while taking various considerations into account, such as convergence rate, the frequency of the changes, desired degree of convergence, and so forth.

This paper presents methods that aim to model computer-side handling of speech exemplars as well as convergence behavior. The parameters and steps of the process are based on the behavior observed in convergence occurring in natural speech [4]. The model's flow is described in Algorithm 1, and all of its parameters and their effects are described in Table 1.

Algorithm 1: Phonetic Response

Input : *ASRInput* – recognized user speech
targetPhonemes – convergence features

Output: list of feature vectors with converged values

```
foreach (phoneme in ASRInput) ∈ targetPhonemes do
    feature ← phoneme.associatedFeature
    context ← feature.phoneticContext
    if not matches(phoneme, context) then
        break
    if inRange(phoneme, allowedRange) then
        if poolSize=maxPoolSize then
            deleteOldestExemplar()
        feature.addExemplar(phoneme)
    else
        break
    if toUpdate = 0 then
        method ← feature.calculationMethod
        poolValue ← method.calculate(pool)
        newValue ←
          rate · poolValue + (1 − rate) · feature.value
        threshold ← convergenceLimit · poolValue
        if newValue > threshold then
            newValue ← threshold
        feature.value ← newValue
        toUpdate ← updatefrequency
    else
        toUpdate ← toUpdate - 1
end
```

2. Background and motivation

Up to now, customization of spoken dialogue systems (SDSs) has mostly concentrated on the dialogue's flow (via the state management component, as in [5]). More advanced systems are capable of reacting to user input in terms of turn taking and response generation [6] using incremental processing [7]. Yet, while the content and timing of the response may be customized for the user, the speech output would be the same for every user using the same system in a similar configuration. Speech adaptation does exist in the field of text to speech (TTS) synthesis. For instance, a new acoustic model can be created from averaging multiple speech corpora (e.g., as the averaged models described in [8, 9]). However, seeing that such models are averaged on the acoustic level, the outcome will be a voice that sounds – as a whole – like an acoustic average of the other models. If an SDS is to use such a technique for converging its speech to the user, the result will be an imitation of the user's voice. That might discourage users, making them feel like they

Table 1: *Summary of model's parameters.*

Parameter	Description
target phoneme	the phoneme that triggers the feature
phonetic context	the environment in which the phoneme is accepted
allowed range	the value range in which new instances are accepted
exemplar pool size	maximum number of exemplars in memory
update frequency	how frequently a feature's value is recalculated
calculation method	the manner in which the pool value is calculated
convergence rate	weight given to the pool value
convergence limit	the maximum degree of convergence allowed for a feature

are talking to themselves or that the system is mocking them. To maintain the system's voice and still match its speech to the user, a more selective approach can be used, where segment-level characteristic of the user's speech are modeled. This results in a system converging to the user's speech properties (like dialectal differences, prosody, vowel quality, etc.), but in its own voice. Given that such convergence occurs in HHI [2, 10], such a system could ultimately lead to a dialogue that resembles natural behavior and is therefore easier to understand and to interact with.

3. Model

As described in Algorithm 1, the response process consists of several steps that are executed sequentially. These steps are based on behavior observed in HHI. Some steps are optional depending on their input, and others terminate the entire process if a condition is not met. The first step is triggered whenever one of the *target phonemes* (configurable) is detected by the speech recognition component. Once such a phoneme is detected, its *context* is examined. This step is necessary to verify that the feature is captured only where its phonological rule is applicable. For example, elision of [ə] occurs only in some specific contexts in German, as described in the following phonological rule (simplified version adapted from [11, pp. 142–143]):

$$\text{ə} \longrightarrow \varnothing \,/\, [\text{-son}] \_\_ \{\#, [\text{+const}]\} \tag{1}$$

The environment in which the recognized phoneme sequence was found is compared to the one defined for the feature associated with the detected phoneme. In case a phoneme is associated with more than one characteristic (for example, a vowel can be associated with both a length feature and a quality feature), this comparison – and any necessary subsequent step – is done for each feature separately. These two steps model the potentially varying characteristics of speech. If the context matches, the process continues.

Each feature is measured with a specific *measurement type*, e.g., formants, duration, label, etc. This measurement defines the way the phoneme (and by extension the feature) is evaluated. It also defines the manner in which evaluation (and later also synthesis) is performed externally. As this is done externally, measuring and interpreting a feature's values can easily be changed and explored by language engineers without being dependent on programmers.

A *range of valid values* is also defined for each feature. This range is used to define the values considered reasonable and expected for the specific feature, and helps to filter out any unwanted values like outliers and recognition errors. If the measured value is not within this range, the exemplar is rejected and the process is terminated for this phoneme. For features composed of multiple values (e.g., multiple formants for vowels), a range is defined for each value separately. In that case, it is enough that one of the values is out of range for the whole exemplar to be rejected. Accepted exemplars are stored in the feature's memory.

The feature's memory is modeled by the *exemplar pool*. The pool's size is configurable and common to all features, modeling a listener's temporal memory. Once a feature's pool has reached its size limit n, exemplars will start to be deleted ("forgotten") on a first-in-first-out basis. That means that the model takes only the n newest exemplars of a feature when calculating its pool value (see below). Thus, the larger the pool's size limit is, the longer the time span influencing the process.

After an exemplar is added to a feature's pool, an update of the feature's value may be triggered. Whether and how often this happens is determined by the *update frequency*. If set to 1, an update will occur every time an exemplar is added; if set to 2, every other exemplar, and so on. When set to 0, however, updates will only take place when explicitly requested. This can be useful when, for example, all features are to be updated at the same time, regardless of how many exemplars have been accumulated for each feature. Increasing the update interval means that each new pool value will be affected by more new exemplars, which, depending on the calculation method used (see below), might result in a smoother converging process. Additionally, a longer update interval also means that convergence will generally take longer, since the model's features are not being updated as frequently.

Along with the size of the exemplar pool, another key decision is how these exemplars will be used to calculate the pool value the model will attempt to converge to. This decision is made by the *calculation method*. Each feature can use another calculation method, and the same calculation method is used for all dimensions of the feature's value. The most basic method would be to average over the exemplar pool. This might be too simplistic, for it does not take any property of the exemplars into account, such as the order in which they were acquired or how similar they are to other exemplars or the correct value. One of the more dynamic methods is decaying average, which recursively gives the newest exemplar a different (configurable) weight. The calculation step can be described by:

$$\mathcal{M}'_i = g(\mathcal{M}^{\intercal}_{i\leq m}) \tag{2}$$

where $\mathcal{M}^{\intercal} \in \mathbb{Q}^{m\times n}$ is the transposed exemplar matrix (which comprises vectors of dimension values instead of vectors of exemplar values), g is the chosen calculation method to apply to each vector, $\mathcal{M}' \in \mathbb{Q}^{m\times n}$ is the matrix with the pool values for each dimension of the feature, i is the vector index, and m is the number of rows in $\mathcal{M}^{\intercal}$.

Experimenting with a new calculation method merely requires to implement a function for an input of number array and to refer to it in the model's configuration file.

Before setting the updated value as the feature's value, the balance between the feature's current value and its calculated pool value needs to be set; this is the *convergence rate*. The purpose of this parameter is to model the tendency of a speaker to converge. Having found in previous work that people's convergence in both HHI [4] and HCI [12] varies considerably, this

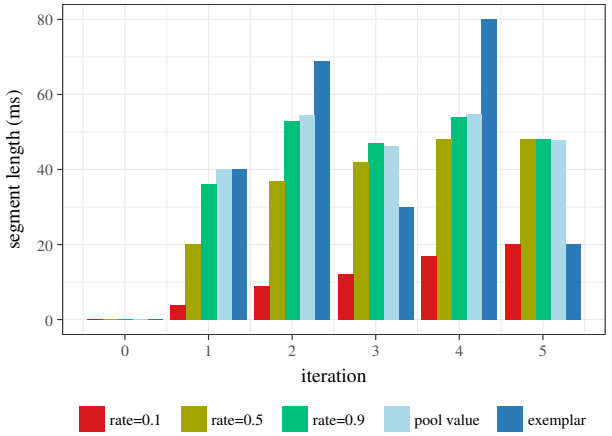

Figure 1: *Comparison between different convergence rates when the calculation method is set to simple average. The "exemplar" bar represents the feature's last exemplar, and the "average" bar represents the accumulated effect of all exemplars at each iteration. The empty first iteration represents the empty* exemplar pool *of the model at the beginning of the process.*

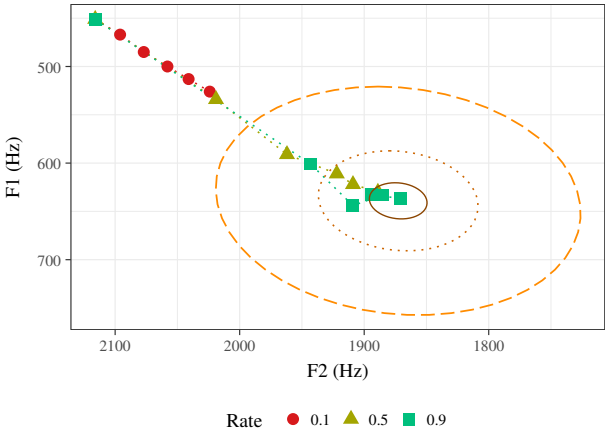

Figure 2: *Comparison between the overall difference between the features' initial states and the observations covered by the model's predictions using different convergence rates using decaying average ($\mu = 0.3$). The points represent the calculation steps (first point is the initial value), and the ellipses represent confidence levels of 90 %, 50 % and 10 %, respectively.*

parameter needs to be tuned with respect to the other parameters to obtain convergence steps that are comparable to natural speech convergence.

A feature's new value is calculated using

$$\Upsilon \equiv \mathcal{C}_u = \rho\upsilon + (1-\rho)\mathcal{C}_{u-1} \tag{3}$$

where Υ is the new feature value (i.e., the value after update u), υ is the calculated pool value from $\mathcal{M}'$ in Equation 6, $\mathcal{C}_{u-1}$ is the current value of the feature (after the previous update), and ρ is the convergence rate. A ρ value of 0 means that no convergence occurs (the current value is retained), a value of 1 will result in complete convergence (the current value is ignored), and $\rho = 0.5$ will result in an average between the two. While the convergence rate is typically a value between 0 and 1, smaller and greater values could be meaningful in some applications to achieve divergence or over-convergence, respectively. For instance, in a tutoring system for pronunciation training, the desired behavior might be for the system to diverge from users' input when it detects that their pronunciation is wrong. By reflecting the users' utterance with diverged pronunciation (instead of explicitly pointing out their mistake), the user receives auditory feedback in a more "conversational" learning process.

Finally, the *convergence limit* defines how close to the input speech the model is allowed to converge. This emulates the speaker's flexibility and to what extent external stimulations can influence their speech. As mentioned above, this flexibility is subject to great variation across speakers. This parameter is important, because it draws the line between a responsive system with its own internal behavior and a system that aims to mimic the user. For example, if set to 0.8, the individual feature (and each of its dimensions) will be limited to 80 % of the new pool value. Hence, in case the new calculated pool value is about to cross this threshold (based on the calculation in Equation 3), it will be set to the maximum allowed value defined by the convergence limit. This value is calculated using

$$\Lambda = \delta\upsilon(1-\lambda) \tag{4}$$

where Λ is the maximum convergence value allowed, δ is set to 1 if the converging values are increasing or -1 in case they are decreasing (see Equation 5), and λ is the convergence limit parameter ($\lambda = 0.8$ in the example above).

Note that the actual value of this limit depends on the direction in which the convergence occurs, which is defined by

$$\delta = \begin{cases} 1 & \text{if } \upsilon \geq \mathcal{C}_t \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

That is, if the converging values are increasing, the limit's value will be smaller than the pool value; and if the values are decreasing toward the pool value, the limit's value will be greater than the pool value.

Ultimately, the final updated value for the feature is determined as follows:

$$\Upsilon = \begin{cases} \upsilon - \Lambda & \text{if } \upsilon - \Upsilon \leq \delta\Lambda \\ \Upsilon \text{ (unchanged)} & \text{otherwise} \end{cases} \tag{6}$$

4. Evaluation

The proposed model was evaluated in two steps. First, the model's predictions were examined and compared with recorded audio (simulating the user's speech input). Secondly, the changes in the target features in the synthesized speech were measured to test the adequacy of the model's output. The speech recognition component of the model integrates CMU Sphinx 4 [13] with a custom language model and dictionary, and the speech synthesis component uses the frequency domain batch filtering[1] functionality of Praat.

The simulated speech used for the evaluation is a subset of the stimuli used in [4]. These stimuli are suitable for simulating convergence, since they were constructed specifically for triggering convergence in human speakers. Three segment-level phonetic features that vary across native speakers of German are incorporated in these stimuli: realization of the word-medial vowel *-ä-* in stressed syllables as [eː] or [ɛː], realization of the word-final sequence *-ig* as [ɪç] or [ɪk], and the elision or epenthesis of [ə] in the word-final sequence *-en*. The former

[1] more details in `http://www.fon.hum.uva.nl/praat/manual/Filtering.html`

Table 2: *Summary of the convergence process for the feature [ɛː] vs. [eː]. Values are the decayed averages ($\mu = 0.3$) at the final step. Coverage is the percentage from the overall difference between the observations and the initial states covered by the model's prediction.*

	Value (Hz)		Coverage	
Rate	F1	F2	F1	F2
0.1	528	2017	39.9 %	35.6 %
0.5	634	1869	94.8 %	88.8 %
0.9	643	1844	99.5 %	97.8 %

two features vary regionally, occurring roughly in the North and South of the German-speaking region of Europe, respectively.

The model features' initial values were always the average of the stimuli (input exemplars) with the opposite feature value. For instance, the feature that captures [ə] elision (see Equation 1) consists of a single value, namely the [ə] segment length. That makes this feature gradual rather than categorical. When letting the system start with segment length of 0 ms (i.e., complete elision) and "listen" to input speech with perceptible [ə] segments of different lengths, the system responds by changing its internal representation of [ə] based on the audio input. This was done to illustrate the significance of the parameter *convergence rate*. Figure 1 summarizes the values of the system in each iteration when using different rates. This figure also illustrates how the value is set based on the entire exemplar pool. For example, when $\rho = 0.5$ (balanced average), the value in the third iteration is larger than its predecessor, even though the exemplar of this iteration introduces a lower value. This is due to the fact that the previous two values were higher, so that the value calculated from the entire pool is still larger than in the second iteration. With $\rho = 0.9$ the changes are much more abrupt, which may lead to an instable and confusing output. With $\rho = 0.1$ the changes are conservative due to the low convergence rate, even though the pool values are much larger. This case represents a speaker which is resistant to external influences. Finally, it is worth pointing out that while the values with $\rho = 0.9$ are closer to the pool value, the change with $\rho = 0.5$ is more gradual and is less likely to overshoot and therefore presents a more natural behavior.

Another phonetic characteristic is a vowel's formants, whose values determine the vowel quality. For example, the orthographic vowel *-ä-* in stressed position could be pronounced as either [ɛː] or [eː]. This characteristic can be ascribed to dialectal or regional influences, and therefore can be subject to convergence when speaking with different interlocutors.

In the first part of the evaluation, the system received input of exemplars where the vowel *-ä-* is pronounced as [ɛː], and set the initial values to those of the exemplars with the pronunciation [eː]: 451 Hz, 2116 Hz, and 2763 Hz for the vowel's F1, F2, and F3 respectively. The rest of the model's parameters (see Table 1) were set as follows:

- exemplar pool size = 10,
- update frequency = 1,
- calculation method = decaying average ($\mu = 0.3$),
- convergence rate = 0.1, 0.5, and 0.9 (for comparisons),
- and convergence limit = 1 (i.e., no limit).

The convergence steps of the the feature [ɛː] vs. [eː] toward the exemplars are shown in Figure 2. Table 2 summarizes the values and convergence distances covered, relative to the averaged exemplar values using different convergence rates. Although the individual covered distance of F3 is larger for $\rho = 0.5$, the overall convergence is superior when $\rho = 0.9$ (since the rate is common to all dimensions of a feature). It may be assumed, then, that a higher convergence rate generally performs better. While that seems to be the case with steady and coherent input, it could result in the undesired behavior of abrupt and frequent, unexpected changes.

Table 3: *Values and errors (from model's predictions) of formant frequencies in audio output sentence using the modified [ɛː] vs. [eː] feature calculated by the model. The values in brackets are the errors compared to the original audio input.*

	Value (Hz)		
Rate	F1	F2	F3
0.1	607	1516	2560
0.5	638	1581	2510
0.9	640	1584	2507
	Error (Hz)		
0.1	79(37)	501(322)	198(203)
0.5	5(6)	288(257)	254(253)
0.9	3(4)	260(254)	260(256)

The second part of the evaluation concentrated on the realization of the predicted values from the first part. For that, some model output was synthesized based on one of the exemplar audio files for the [ɛː] vs. [eː] feature, representing possible speech output of an SDS that uses the model. The formant values in the segment containing the feature in the synthesized audio were manually measured at the vowel's midpoint and compared to the model's prediction. Due to limitations of the used synthesis method, the manipulation did not result in the exact values of the model output (see Table 3). Nevertheless, the vowel realizations changed from [eː] to [ɛː], which means that the end-to-end process successfully generated speech output that responded to the initial speech input.

5. Conclusion and future work

A novel model for capturing and responding to segment-level phenomena was presented and evaluated. Both the model predictions and synthesized output based on them were evaluated, and the results demonstrate the feasibility of the concept and its potential benefit as an SDS component. The model's predictions were found to be up to 99.5 % accurate. The synthesized speech output was not as precise, but the realization of the target phonetic phenomena was nonetheless successfully changed based on the simulated speech input. This is due to limitations of the synthesis method and not the model itself, and is expected to improve when using another method.

This model could, in principal, be extended to capture other characteristics of HCI like prosody and other suprasegmental features, turn-taking, and even lexical decisions, to customize the interaction on other levels as well. Personalized behaviors like these in computers can assist in the research of convergence, since it offers a more controlled experimental environment. Our understanding of convergence in speech from the theoretical point of view may benefit from such experiments.

Future work is planned to integrate the model into an SDS using a framework such as OpenDial [14] or IrisTK [15]. Furthermore, support for a broader range of phonetic phenomena and more suitable TTS engines would also improve the system's speech output.

6. References

[1] J. S. Pardo, "On phonetic convergence during conversational interaction," *Journal of the Acoustical Society of America*, vol. 119, no. 4, pp. 2382–2393, Apr. 2006.

[2] N. Lewandowski, "Talent in nonnative phonetic convergence," Ph.D. dissertation, Universität Stuttgart, 2012.

[3] J. S. Pardo, I. C. Jay, and R. M. Krauss, "Conversational role influences speech imitation," *Attention, Perception, & Psychophysics*, vol. 72, no. 8, pp. 2254–2264, Nov. 2010.

[4] I. Gessinger, E. Raveh, J. O'Mahony, I. Steiner, and B. Möbius, "A shadowing experiment with natural and synthetic stimuli," in *Phonetik & Phonologie 12*, Oct. 2016, pp. 58–61.

[5] A. N. Pargellis, H.-K. J. Kuo, and C.-H. Lee, "An automatic dialogue generation platform for personalized dialogue applications," *Speech Communication*, vol. 42, no. 3-4, pp. 329–351, Apr. 2004.

[6] H. Buschmeier, T. Baumann, B. Dosch, S. Kopp, and D. Schlangen, "Combining incremental language generation and incremental speech synthesis for adaptive information presentation," in *13th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, Jul. 2012, pp. 295–303.

[7] T. Baumann and D. Schlangen, "The INPROTK 2012 release," in *NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data*, Jun. 2012, pp. 29–32.

[8] R. Dall, C. Veaux, J. Yamagishi, and S. King, "Analysis of speaker clustering strategies for HMM-based speech synthesis." in *Interspeech*, Sep. 2012, pp. 995–998.

[9] C. Veaux, J. Yamagishi, and S. King, "Using HMM-based speech synthesis to reconstruct the voice of individuals with degenerative speech disorders." in *Interspeech*, Sep. 2012, pp. 967–970.

[10] M. Babel, G. McGuire, S. Walters, and A. Nicholls, "Novelty and social preference in phonetic accommodation," *Laboratory Phonology*, vol. 5, no. 1, pp. 123–150, Jan. 2014.

[11] W. A. Benware, *Phonetics and Phonology of Modern German: An Introduction*. Georgetown University Press, 1986.

[12] E. Raveh, I. Gessinger, S. Le Maguer, B. Möbius, and I. Steiner, "Investigating phonetic convergence in a shadowing experiment with synthetic stimuli," in *28th Conference on Electronic Speech Signal Processing (ESSV)*, Mar. 2017, pp. 254–261.

[13] P. Lamere, P. Kwok, E. Gouvea, B. Raj, R. Singh, W. Walker, M. Warmuth, and P. Wolf, "The CMU SPHINX-4 speech recognition system," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, Apr. 2003, pp. 2–5.

[14] P. Lison and C. Kennington, "OpenDial: A toolkit for developing spoken dialogue systems with probabilistic rules," in *54th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Aug. 2016, pp. 67–72.

[15] G. Skantze and S. Al Moubayed, "IrisTK: a statechart-based toolkit for multi-party face-to-face interaction," in *14th ACM International Conference on Multimodal interaction (ICMI)*, Oct. 2012, pp. 69–76.