INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Timing Generating Networks: Neural Network based Precise Turn-taking Timing Prediction in Multiparty Conversation

Shinya Fujie[1,2], *Hayato Katayama*[2], *Jin Sakuma*[2], *Tetsunori Kobayashi*[2]

[1]Chiba Institute of Technology, Japan
[2]Waseda University, Japan
shinya.fujie@p.chibakoudai.jp

Abstract

A brand new neural network based precise timing generation framework, named the Timing Generating Network (TGN), is proposed and applied to turn-taking timing decision problems. Although turn-taking problems have conventionally been formalized as users' end-of-turn detection, this approach cannot estimate the precise timing at which a spoken dialogue system should take a turn to start its utterance. Since several conventional approaches estimate precise timings but the estimation executed only at/after the end of preceding user's utterance, they highly depend on the accuracy of intermediate decision modules, such as voice activity detection, etc. The advantages of the TGN are that its parameters are tunable via error backpropagation as it is described in a differentiable form as a whole, and it is free from inter-module error propagation as it has no deterministic intermediate modules. The experimental results show that the proposed system is superior to a conventional turn-taking system that adopts the hard decisions on user's voice activity detection and response time estimation.

Index Terms: spoken dialogue system, turn taking, timing control

1. Introduction

In this paper, we propose a novel framework of a neural network-based timing detection system that decides the spoken dialogue system's utterance initiation timing by implementing an adjustable timer while changing its time constant depending on the conversational context.

The turn-taking problem has conventionally been formalized as users' end-of-turn detection based on acoustic and linguistic features[1, 2]. However, precise timing control of turn-taking has seldom been considered. Conventional approaches of turn-taking typically assume systems can naïvely take their turns immediately right after users release them, or set timers to start speaking after specific periods, though pause duration before the next utterance initiation is not constant at all. [3, 4, 5, 6, 7, 8, 9, 10] attempted to improve the accuracy of sequential end-of-turn estimation, and [11, 12, 13, 14, 15, 16, 9] formalized the end-of-turn detection as a classification problem, whether or not taking a turn for each end of utterance. Still, these approaches focus solely on estimating when the interlocutor releases a turn, assuming that a system can take a turn immediately when the interlocutor releases it. Nevertheless, neither responding as quickly as possible nor waiting for fixed time is always appropriate. It depends on dialogue context how long a system should wait. While the system should respond quickly after the interlocutor explicitly gives his/her turn to it, it should wait for a while when it is ambiguous who should speak next so that it prevents from interfering other participants' utterances. Even though recent end-to-end neural networks have revolutionized the field of conversational systems by mitigating the inter-module error propagation that conventional symbolic turn-taking systems typically cause, they still lack precise time structures capable enough to handle sophisticated turn-taking.

The TGN continuously estimates degree of appropriateness of every single moment as a target timing, and detects the timing when the estimated value exceeds the threshold. In turn-taking timing detection, the value is defined as the utterance expectation, which is how much other participants expect the system to speak. In the training of the TGN, the parameters in the network are optimized as the estimated utterance expectation crosses over the threshold just at the system's turn-taking timing in the corpus. To realize such a training with the limited number of the correct timings, the time function of utterance expectation is modeled with restricted form, which increases during the participants remain silent. The TGN mainly estimates the increasing speed of the utterance expectation. In this study, we integrate response obligation recognition as another task to improve the accuracy of the estimation.

The following section describes the TGN architecture, section 3 reports the experiments, section 4 discusses the experimental results, and section 5 concludes with future directions.

2. Timing Generating Networks

2.1. Utterance Expectation

TGN continuously estimates the utterance expectation, which means how much other participants expect the system to speak, and detects the turn-taking timing when this value exceeds the threshold. Although it is vague whether the system should take its turn or not immediately after the end of the preceding utterance, it becomes clearer as time goes on with other participants remaining in silent. Thus, the estimation of utterance expectation is modeled as increasing with silent time. Moreover, since the timing of turn-taking changes depending on context of conversation, the speed of increase in utterance expectation should be estimated with the context.

Therefore, we model estimated the value of utterance expectation as

$$y_t = \alpha u_t + (1 - \alpha) y_{t-1} \tag{1}$$

where t denotes discrete time index, u_t is the degree of silence, and α is a parameter. This equation means it is modeled as IIR filter which is a first order lag system where u_t is the input, y_t is the output. α $(0 \leq \alpha \leq 1)$ is the parameter to control how quickly y_t responds to u_t.

2.2. Timing Generating Network Framework

The overall architecture of the TGN is shown in Fig. 1. It consists of three components: Filter (FLT), Response Controller (CTR), and non-Speech Detector (NSD).

http://dx.doi.org/10.21437/Interspeech.2021-874

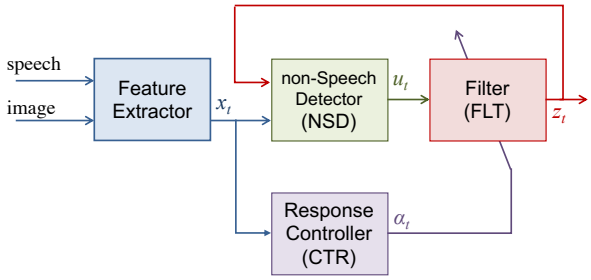


Figure 1: *Block diagram of the TGN consisting of CTR, NSD and FLT components*

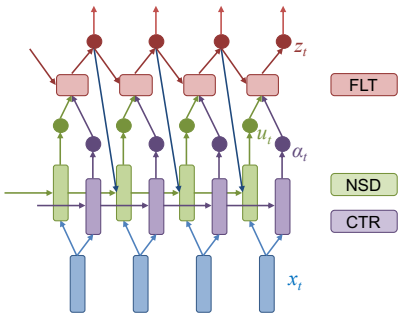


Figure 2: *Network structure of the TGN. Cells of purple, green, and red represent CTR, NSD and FLT respectively.*

FLT is the module to calculate the utterance expectation y_t by (1). When y_t exceeds a certain threshold, it is judged to be the utterance initiation timing for the system. Namely, it set the z_t, the command to start utterance, to 1.

$$z_t = \begin{cases} 1 & (y_t \geq threshold) \\ 0 & (otherwise) \end{cases}$$

y_t is reset to 0 when the system or user starts another utterance.

CTR is the module that determines parameters of FLT depending on the context. In this study, since FLT has only a single parameter, α, the output is a scalar. This part is composed with LSTM layer and the output is calculated using multi-modal feature sequence. Since the output is incrementally calculated, it is not a constant but a time-variant variable α_t. To regard it as the speed of increase in utterance expectation, it should be less varied while no one speaks and the expectation is increasing. Thus, we adopts the following gate mechanism

$$\alpha'_t = u_t \alpha'_{t-1} + (1 - u_t)\alpha_t$$

and treat α'_t as the output of the CTR.

NSD determines u_t that is the input of FLT. Since u_t is the degree of user's silence, it calculates the posterior probability of user's voice activity using LSTM based voice activity detector, then invert and output it.

Figure 2 shows the TGN network structure. This system is designed in a differentiable form that can be applied to end-to-end learning. That is, all the processes are not deterministic, therefore each module's errors don't critically damage the overall performance. All the modules are interdependent and exchange information synchronously with analytic frames.

2.3. Feature Extraction

We consider multi-modal features including acoustic, visual, and linguistic features to model the utterance timing of the multiparty conversation system.

We assume the acoustic features of each user are dominant to determine the likelihood of user's silence. In our previous work, we extracted the acoustic features using the same approach[17]. Following the method, we extract a narrow-band spectrogram of the audio signal and use the bottleneck features (256 dimensions) of the middle layer when building CNN AutoEncoder.

As the visual features, user's image information obtained from the system camera is used, based on our assumption that the information of user's gaze and face directions is essential for the addressee identification[18]. The visual features are extracted by detecting faces and cutting out the upper third of the eyes with dlib[1][19]. In addition, we constructed a gaze estimator from the data we collected separately, and use the middle layer as the bottleneck features (64 dimensions).

As the linguistic features, we use the output vector of the intermediate layer of the CTC-based phoneme recognition system. Although using results of large vocabulary speech recognition system is better from the point of view of the accuracy, it needs post-processing such as word embedding etc. and not suitable for real time processing. The phoneme recognition model was pre-trained with the CSJ(Corpus of Spontaneous Japanese) and its input feature is the same as the aforementioned acoustic features.

2.4. Optimization

Algorithm 1 shows a pseudo code of the optimization flow of utterance timing of the system by the TGN. When the set of times at which the system starts to speak is C, the system optimizes the utterance expectation y_t to the threshold defined by the system at time $t(\in C)$. The training phase is largely composed of two steps. The first step is the optimization of the NSD block. Only the parameters $\boldsymbol{\theta}_u$, that estimate user's silence likelihood u_t, are trained first. The second step is the optimization of the CTR and the FLT blocks. The parameters $\boldsymbol{\theta}_u$ are not updated when training $\boldsymbol{\theta}_\alpha$. The utterance expectation of the system y_t is calculated by forward propagation.

Losses are then calculated at specific timings described as follows: First, at the timing of the system's actual utterance, $t\ (\in C)$, the loss is calculated so that y_t is closer to T_1. However, if the system is trained only at the timing $t\ (\in C)$, the system cannot avoid the phenomenon of y_t becoming unnecessarily large. Therefore, at any timing where y_t exceeds T_1 other than the appropriate timing, the difference between $T_2\ (< T_1)$ and y_t is calculated and added to the loss. Minimization of it is expected to suppress the increase in y_t and prevents it from crossing the threshold at inappropriate timing. Note that loss is computed only for the frame that becomes $y_t \geq T_1$ in any case, and no computation is performed until $y_t < T_1$ again in subsequent intervals of $y_t \geq T_1$.

The labels used for training are very sparse data that have values only at the beginning of the system's utterance. Because we compute the loss at specific frames, this model is trainable for sparse data.

[1] http://dlib.net/

Algorithm 1 Optimizing utterance timing by the TGN

Require:
- Set of system turn taking timing C,
- Input x_t,
- u_t target v_t,
- Target T_1, T_2,
- Number of samples N,

Ensure:
- Neural network parameter $\{\boldsymbol{\theta}_\alpha, \boldsymbol{\theta}_u\}$
- Output $\{u_t, \alpha_t, y_t\}$,

// **step 1**. training NSD block (parameter $\boldsymbol{\theta}_u$),
for t from 1 to N **do**
  $u_t \leftarrow LSTM(x_t; \boldsymbol{\theta}_u)$
  $L_u \leftarrow$ softmax_crossentropy(u_t, v_t)
  update $\boldsymbol{\theta}_u$ with L_u
end for
// **step 2**. training CTR block (parameter $\boldsymbol{\theta}_\alpha$),
for t from 1 to N **do**
  $u_t \leftarrow LSTM(x_t; \boldsymbol{\theta}_u)$
  $\alpha_t \leftarrow LSTM(x_t; \boldsymbol{\theta}_\alpha)$
  $\alpha'_t \leftarrow u_t \alpha'_{t-1} + (1 - u_t)\alpha_t$
  $y_t \leftarrow \alpha'_t u_t + (1 - \alpha'_t) y_{t-1}$
  if $t \in C$ **then**
    $L_\alpha \leftarrow$ square_error(y_t, T_1)
  end if
  if $y_t \geq T_1$ (only the first timing t exceeding T_1) **then**
    $L_\alpha \leftarrow$ square_error(y_t, T_2)
  end if
  update $\boldsymbol{\theta}_\alpha$ with L_α
end for

2.5. Integrating Response Obligation Recognition

We consider "response obligation" in calculation of the utterance expectation. Here, the response obligation is assumed to be recognized mainly with the linguistic contents of the conversation. For example, in the case the preceding utterance is a question to another participant which means an explicit request to take the turn, the response obligation exists on the participant. In most of the cases which obligation exists on the system, the system should take its turn quickly.

We integrate response obligation recognition (ROR) as a subtask of TGN. As shown in Figure. 3, with intermediate output of the CTR, the ROR module recognizes the response obligation. In the optimization of ROR integrated TGN, ROR loss is calculated with softmax cross entropy using the annotated labels (RO exists or not) at the end of user's utterance.

3. Experiment

3.1. Experimental Setting

To verify the effectiveness of the TGN, we first collected multiparty conversation data. The detailed design of the data collection is described in [20]. To mitigate annotation costs, we employed the conventional Wizard of OZ (WoZ) method[21].

We used SOTA[2], a commercially available robot, as the experimental platform. Each participant was asked to join by pairing with his/her close friend and has a conversation to decide what movies they would like to watch based on the information provided by the robot. Because of the close relationship of each pair, the conversational flows were expected to be natural.

[2] https://www.vstone.co.jp/products/sota/

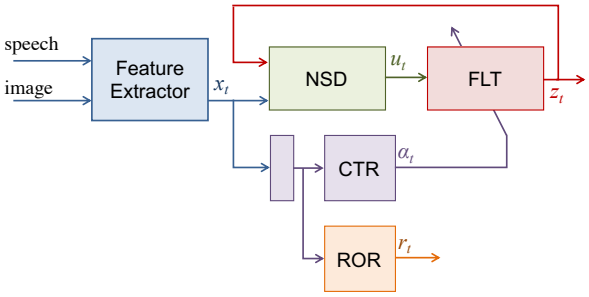


Figure 3: *TGN integrated with ROR(Response Obligation Recognition)*

Totally 40 men and women averaged around twenties participated in the data collection, and each session lasted about three to ten minutes. The speech of each participant was collected via each pin microphone on his/her chest, and the facial information was captured via the robot's head-mounted camera (the operator controlled the robot as it looks at the current speaker's face). Totally 110 sessions, approximately 10 hours of conversation, were recorded.

z_t was given as the timing of the correct answer, which is the timing when the conversation system actually begins to speak (this label is annotated at the time of data collection using the WoZ method). y_t at the timing of the correct answer was set to 0.8 ($T_1 = 0.8$ in Algorithm 1). When this frame overlaps with the interval of $u_t = 0$, the correct answer timing is delayed to the next frame of u_t becomes 1. We also set a target value of 0.2 to penalize those that exceed this value at the wrong time ($T_2 = 0.2$ in Algorithm 1).

To compare with the TGN, we conducted experiments with two conventional methods, Pause-Based method (PB) and Response Timing Generator (RTG). PB simply estimates the pause, how long system should wait to take its turn, at the end of the preceding utterance. It is implemented in the neural network architecture similar to the TGN but it directly estimates pause duration. RTG is based on the approach proposed in [22]. It classifies whether the system should take its turn or not at every 50ms step after the end of the preceding utterance is detected. The difference from the method in [22] is that it is re-implemented with neural network and uses the same extracted feature as TGN.

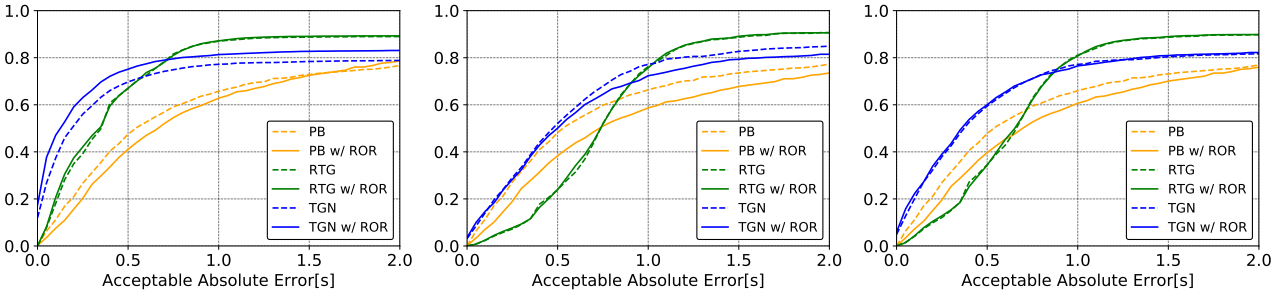
3.2. Result

The experimental results are shown in Table 1 and Figure 4. The results are precision, recall, F_1 score calculated with variation of acceptable absolute errors between the detected timings and the correct timings. In the case the error is 0.4s the proposed method (TGN) is superior to other methods. On the other hand, when the error is more than 1.0s, RTG is better. ROR contributes the improvement especially in precision of the TGN in the small error range.

Figure 5 shows the histograms of estimated α at the ends of preceding utterances. It is confirmed that with integrating ROR the estimated α is distinguished more clearly depending on the existence of response obligation.

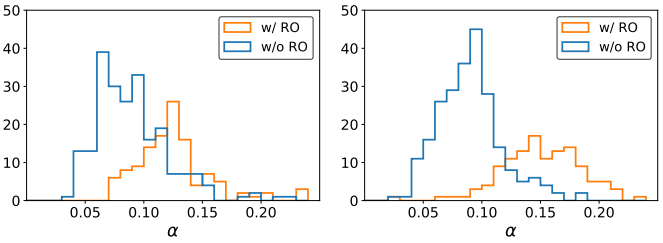
Table 1: *Experimental Results.* P_x, R_x, *and* F_x *indicate precision, recall, and* F_1 *score, respectively, calculated as* x *seconds are acceptable absolute errors. ROR Acc. indicates the accuracy of response obligation recognition.*

Method	$P_{0.4}$	$R_{0.4}$	$F_{0.4}$	$P_{0.8}$	$R_{0.8}$	$F_{0.8}$	$P_{1.6}$	$R_{1.6}$	$F_{1.6}$	**ROR Acc.**
PB	0.403	0.410	0.406	0.605	0.611	0.608	0.737	0.743	0.739	—
PB *w/* ROR	0.336	0.314	0.325	0.566	0.529	0.547	0.736	0.688	0.711	0.871
RTG	0.608	0.178	0.271	**0.833**	0.584	0.681	0.888	0.894	0.891	—
RTG *w/* ROR	0.595	0.164	0.252	0.832	0.583	0.677	**0.892**	**0.898**	**0.894**	0.850
TGN	0.661	0.440	0.526	0.756	**0.705**	**0.729**	0.785	0.833	0.808	—
TGN *w/* ROR	**0.721**	**0.429**	**0.537**	0.801	0.668	0.728	0.828	0.798	0.812	0.852



(a) Precision (b) Recall (c) F_1

Figure 4: *Evaluation variation with change in acceptable absolute errors*



(a) TGN (*w/o* ROR) (b) TGN *w/* ROR

Figure 5: *Histograms of estimated* α

4. Discussion

The experimental results show that the TGN performs well in the small error range. According to results of our subjective preliminary experiments, it was revealed that turn taking timing errors shorter than 0.5s are acceptable. Therefore, for the purpose of maintaining the rhythm of conversation, TGN (especially with ROR) is the most suitable since it achieved the best accuracy in the 0.5s error range.

TGN has no intermediate deterministic module, and it can quickly detect a timing immediately after u_t increases if α is estimated as a large value. On the other hand, the both of comparison methods have end-of-utterance detection as an intermediate deterministic module. Since they must wait for the certain silent period to prevent the error of the end-of-utterance detection, detection of turn-taking timing also delays accordingly. Therefore, especially in the cases that the system should take its turn immediately after a preceding utterance finishes, the TGN could detect the timing more correctly.

Such an advantage of the TGN is accelerated with ROR. Figure 5 shows that TGN with ROR generates larger α (> 0.17) than without ROR, as well as it generates clearly distinguished values depending on RO. That implies integration of ROR makes turn-taking timing faster and contributes improvement in precision.

5. Conclusion

In this paper, to address the problem of the turn initiation timing control based on the user's utterance expectation, we proposed the Timing Generating Network (TGN) by implementing an adjustable timer while changing its speed depending the conversational context. We then applied it to a multiparty conversation robot system that can decide its turn initiation timing. The experiment results showed the proposed approach outperforms conventional methods especially in small acceptable error timing range. To supplement the quantitative experiment described in this paper, future work includes subjective experiments to evaluate the overall impression of the proposed system.

6. Acknowledgments

The research was supported by NII CRIS collaborative research program operated by NII CRIS and LINE Corporation.

7. References

[1] A. Raux and M. Eskenazi, "A finite-state turn-taking model for spoken dialog systems," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2009, pp. 629–637.

[2] A. Gravano and J. Hirschberg, "Turn-yielding cues in task-

oriented dialogue," in *Proceedings of the SIGDIAL 2009*, 2009, pp. 253–261.

[3] M. Henderson, B. Thomson, and S. Young, "Word-based dialog state tracking with recurrent neural networks," in *Proceedings of SIGDIAL 2014*, Jun. 2014, pp. 292–299.

[4] R. Smith, "Comparative error analysis of dialog state tracking," in *Proceedings of SIGDIAL 2014*, Jun. 2014, pp. 300–309.

[5] K. Sun, L. Chen, S. Zhu, and K. Yu, "The SJTU system for dialog state tracking challenge 2," in *Proceedings of the SIGDIAL 2014*, Jun. 2014, pp. 318–326.

[6] B.-J. Lee, W. Lim, D. Kim, and K.-E. Kim, "Optimizing generative dialog state tracker via cascading gradient descent," in *Proceedings of SIGDIAL 2014*, Jun. 2014, pp. 273–281.

[7] N. G. Ward and D. DeVault, "Ten challenges in highly-interactive dialog systems," in *Proceedings of AAAI 2015 Spring Symposium*, Palo Alto, CA, Mar. 2015.

[8] O. Plátek, P. Belohlávek, V. Hudecek, and F. Jurcícek, "Recurrent neural networks for dialogue state tracking," *CoRR*, vol. abs/1606.08733, 2016. [Online]. Available: http://arxiv.org/abs/1606.08733

[9] R. Masumura, T. Tanaka, A. Ando, R. Ishii, R. Higashinaka, and Y. Aono, "Neural dialogue context online end-of-turn detection," in *Proceedings of SIGDIAL 2018*, Jul. 2018, pp. 224–228.

[10] Z. Aldeneh, D. Dimitriadis, and E. M. Provost, "Improving end-of-turn detection in spoken dialogues by detecting speaker intentions as a secondary task," *CoRR*, vol. abs/1805.06511, 2018. [Online]. Available: http://arxiv.org/abs/1805.06511

[11] M. Shannon, G. Simko, S. yiin Chang, and C. Parada, "Improved end-of-query detection for streaming speech recognition," in *Proceedings of Interspeech 2017*, Aug. 2017, pp. 1909–1913.

[12] A. Maier, J. Hough, and D. Schlangen, "Towards deep end-of-turn prediction for situated spoken dialogue systems," in *Proceedings of Interspeech 2017*, 08 2017, pp. 1676–1680.

[13] M. Roddy, G. Skantze, and N. Harte, "Investigating speech features for continuous turn-taking prediction using LSTMs," *CoRR*, vol. abs/1806.11461, 2018. [Online]. Available: http://arxiv.org/abs/1806.11461

[14] ——, "Multimodal continuous turn-taking prediction using multiscale RNNs," *CoRR*, vol. abs/1808.10785, 2018. [Online]. Available: http://arxiv.org/abs/1808.10785

[15] K. Hara, K. Inoue, K. Takanashi, and T. Kawahara, "Prediction of turn-taking using multitask learning with prediction of backchannels and fillers," in *Proceedings of Interspeech 2018*, Sept. 2018, pp. 991–995.

[16] D. Lala, K. Inoue, and T. Kawahara, "Evaluation of real-time deep learning turn-taking models for multiple dialogue scenarios," *Proceedings of ICMI 2018*, pp. 78–86, 2018.

[17] K. Yokoyama, H. Takatsu, H. Honda, S. Fujie, and T. Kobayashi, "Investigation of users' short responses in actual conversation system and automatic recognition of their intentions," in *Proceedings of SLT 2018*, Feb. 2019, pp. 934–940.

[18] N. Jovanovic, R. op den Akker, and A. Nijholt, "Addressee identification in face-to-face meetings," in *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006, pp. 169–176.

[19] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, no. 60, pp. 1755–1758, 2009. [Online]. Available: http://jmlr.org/papers/v10/king09a.html

[20] H. Katayama, S. Fujie, and T. Kobayashi, "End-to-middle training based action generation for multi-party conversation robot," in *Proceedings of IWSDS 2019*, Apr. 2019.

[21] N. Dahlböck, A. Jönsson, and L. Ahrenberg, "Wizard of Oz studies—why and how," *Knowledge-Based Systems*, vol. 6, no. 4, pp. 258 – 266, 1993, special Issue: Intelligent User Interfaces.

[22] M. Takeuchi, N. Kitaoka, and S. Nakagawa, "Generation of natural response timing using decision tree based on prosodic and linguistic information," in *Proceedings of Eurospeech 2003*, 2003, pp. 609–612.