Beyond Unified Models: A Service-Oriented Approach to Low Latency, Context Aware Phonemization for Real Time TTS

Mahta Fetrat Qharabagh, Donya Navabi, Zahra Dehghanian, Morteza Abolghasemi, Hamid R. Rabiee

Sharif University of Technology / Tehran, Iran

Correspondence: rabiee@sharif.edu

Abstract

Lightweight, real-time text-to-speech systems are crucial for accessibility. However, the most efficient TTS models often rely on lightweight phonemizers that struggle with context-dependent challenges. In contrast, more advanced phonemizers with a deeper linguistic understanding typically incur high computational costs, which prevents real-time performance.

This paper examines the trade-off between phonemization quality and inference speed in G2P-aided TTS systems, introducing a practical framework to bridge this gap. We propose lightweight strategies for context-aware phonemization and a service-oriented TTS architecture that executes these modules as independent services. This design decouples heavy context-aware components from the core TTS engine, effectively breaking the latency barrier and enabling real-time use of high-quality phonemization models. Experimental results confirm that the proposed system improves pronunciation soundness and linguistic accuracy while maintaining real-time responsiveness, making it well-suited for offline and end-device TTS applications.

1 Introduction

Text-to-speech (TTS) conversion is a long-established and well-developed task, with a wide range of approaches and architectures proposed over the years. The choice or design of a particular TTS method today depends largely on the specific needs and requirements of the application.

One essential use case for TTS is in screen readers, where the system must operate in real-time, offline, on low-end hardware devices. Users in this setting are exposed to the synthesized voice for long periods every day, so the output must not sound robotic or unpleasant. This scenario imposes three main requirements on the TTS engine: 1) Lightweightness, 2) Real-time performance, and 3) Naturalness.

Unfortunately, there is a clear trade-off among these requirements. Larger and more complex neural models often produce highly natural, human-like speech but require significantly more computational resources and introduce higher inference latency. Conversely, smaller neural models, or traditional rule-based, non-neural systems, are much faster and lighter but lack the capacity to model smooth, natural-sounding human speech.

Simply reducing model size to meet speed and lightweight requirements often degrades speech naturalness. Many recent systems, however, maintain acceptable naturalness by decoupling grapheme-to-phoneme (G2P) conversion from phoneme-to-speech (P2S) synthesis (OHF-Voice, 2025; Mehta et al., 2024; Li et al., 2025). Instead of learning an end-to-end text-to-speech mapping, these systems first convert text to phonemes using a lightweight G2P module, then generate speech from the phoneme sequence with a neural synthesizer. This allows the neural component to focus on a narrower task, enabling smaller models and faster inference while preserving reasonable quality.

However, this decoupling makes the overall naturalness and intelligibility of the output heavily dependent on the performance of the G2P module, a task that remains highly challenging for languages with complex or ambiguous phonemization rules.

For example, in Persian, many cases require context-aware phonemization. Two major challenges are:

1. Homographs, i.e., words with multiple valid pronunciations depending on context (e.g., the English word *read*, pronounced either /riːd/ or /rɛd/ depending on tense), and

2. The Ezafe phoneme, a connecting /e/ sound that appears between grammatically or seman-

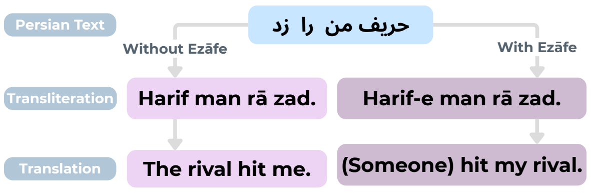

Figure 1: An example of how the Persian Ezafe phoneme (/e/) can change the meaning of a sentence.

tically related words, again determined by context.

Figure 1 illustrates how the presence or absence of a single Ezafe phoneme can alter the meaning of a sentence, highlighting the importance of correctly determining it based on context.

Highly non-phonetic and ambiguous languages pose a challenge for lightweight, real-time TTS systems. While embedding a strong, context-aware G2P model could greatly improve pronunciation soundness and correctness, such models are typically large neural networks, and integrating them directly would compromise speed and efficiency. Existing lightweight TTS architectures decouple G2P from P2S, but their G2P modules remain limited for ambiguous languages. Enhancing these modules with context-aware neural models introduces the very latency and computational overhead that lightweight TTS aims to avoid. This is the central challenge addressed in this paper.

In this paper, we propose a method to overcome the latency barrier for incorporating context-aware phonemizers into real-time TTS systems. Our approach combines two complementary strategies: lightweight, statistically driven modules that provide partial context-awareness, and a service-oriented architecture that allows heavier neural phonemizers to run independently, without embedding them directly in the TTS runtime. The core idea is to move beyond the traditional unified TTS design by treating utility modules as independent services, which the main TTS engine can query as needed, avoiding their computational and loading overhead.

Key contributions of this work are as follows:

1. Proposing a service-oriented approach for integrating neural components into real-time TTS systems,

2. Presenting a service-oriented adaptation of the well-known PiperTTS architecture,

3. Introducing a lightweight, fast, and context-aware phonemizer tailored to Persian phonemization challenges, an enhanced version of the existing eSpeak phonemizer, and

4. Providing a new Persian voice for Piper, trained on the largest publicly available Persian TTS dataset to date.

2 Related Works

2.1 TTS approaches

TTS is a longstanding task that has been in existence since 1939 (Dudley et al., 1939). It began with rule-based methods that utilized handwritten pronunciation and prosody rules, along with simple formant/articulatory synthesis, to generate speech (Klatt, 1980, 1987). Then it proceeded to the next generation, utilizing concatenative unit-selection (Sagisaka, 1988; Hunt and Black, 1996; Black and Taylor, 1997) and later statistical parametric systems (e.g., HMM-based acoustic models with vocoders) (Tokuda et al., 2000; Zen et al., 2009), which improved stability and footprint but still had limitations in terms of naturalness. Like many other tasks, it then evolved into deep-learning-based methods like sequence-to-sequence acoustic models (Tacotron-style) (Wang et al., 2017; Shen et al., 2018) paired with neural vocoders (WaveNet/flow/GAN) (Van Den Oord et al., 2016; Prenger et al., 2019; Yamamoto et al., 2020) and fully end-to-end models such as VITS (Kim et al., 2021) and non-autoregressive FastSpeech-style models (Ren et al., 2019, 2020); these can be grouped by architecture families (autoregressive, non-autoregressive, flow-based, diffusion-based) (Kim et al., 2020; Popov et al., 2021; Kim et al., 2022; Mehta et al., 2024). And most recently, it is performed by large language models, such as commercial or research foundation TTS systems (e.g., VALL-E/Bark-style and TTS components integrated into general LLM stacks) (Wang et al., 2023; Le et al., 2023), which offer strong quality but usually require online GPU inference.

In this paper, our focus is on TTS architectures suitable for offline, real-time, end-device applications. Therefore, we limit our discussion to models that can operate efficiently in CPU-first, low-latency settings.

In practice, we can narrow our focus to TTS architectures that include a distinct phonemization

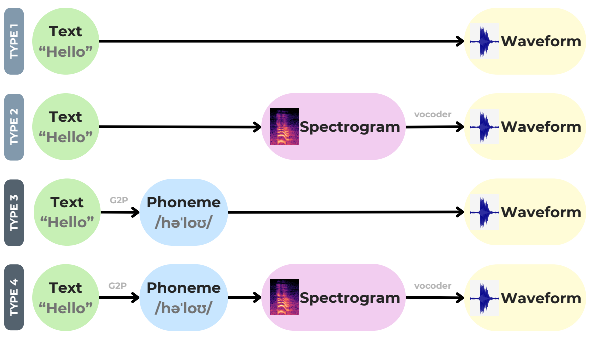

Figure 2: Four common granularity levels in TTS architectures, differing by intermediate representations.

stage. Broadly, modern TTS systems can be categorized into four levels of architectural granularity (Figure 2). At one extreme, fully end-to-end models map raw text directly to waveform; at the other, highly modular pipelines decompose the task into explicit stages: first converting text to phonemes, then generating a spectrogram, and finally synthesizing the waveform through a vocoder. Understanding the degree of granularity in a given TTS architecture provides insight into the complexity of the problem the model must solve, the capacity it may require, and the latency implications of its intermediate components. This perspective is essential when selecting an appropriate model for applications with constraints such as low-latency or limited compute.

VITS, FastSpeech-family models, Glow-TTS, and other flow-based systems, as well as recent diffusion/consistency approaches (e.g., Matcha-like designs), are the most relevant to our goals (Kim et al., 2021; Ren et al., 2019, 2020; Kim et al., 2020; Mehta et al., 2024; Rhasspy Team, 2023).In summary: VITS merges acoustic modeling and vocoding and offers good quality with low-latency; FastSpeech generates spectrograms in parallel and is very fast with a light vocoder; flow-based models enable stable alignments and parallel inference; diffusion/consistency models improve robustness and quality with careful inference schedules (Kim et al., 2021; Ren et al., 2019; Kim et al., 2020; Mehta et al., 2024; Li et al., 2023).

Piper architecture, which is the baseline model in this study, is closely related to these families and is based on VITS with practical improvements for deployment (ZachB100, 2023). It uses a modular structure with a G2P front-end and a phoneme-to-speech (P2S) neural back-end. By moving the G2P step outside the neural model (typically using a lightweight rule-based phonemizer such as eSpeak-ng (eSpeak NG developers, 2013)) and exporting models to ONNX for CPU-friendly inference, Piper reduces model size, and improves speed while maintaining acceptable naturalness. For a more detailed justification for choosing Piper as our baseline, please refer to Appendix A.

2.2 G2P Tools

Since a central focus of this study is enhancing the G2P component of a TTS system, we briefly review the relevant literature and available tools.

G2P methods have evolved in parallel with TTS systems. Early rule-based approaches and pronunciation lexicons were compact and predictable but struggled with out-of-vocabulary words and context-dependent pronunciations. Statistical methods such as finite-state and n-gram letter-to-sound models and CRF-based taggers generalized better while remaining relatively lightweight (Beesley and Karttunen, 2003; Bisani and Ney, 2008; Jiampojamarn et al., 2007). Neural models, including RNNs and Transformers, have since achieved state-of-the-art accuracy by capturing longer-range dependencies (Yao and Zweig, 2015; Vaswani et al., 2017), but they typically require more compute and memory than lightweight statistical or rule-based methods (Park, 2019).

In the case of Persian, several non-scholarly G2P implementations exist on platforms such as GitHub (Dehghani, 2022; Pascal, 2020; Rabiee, 2019; Ajini, 2022; Mortensen et al., 2018; Alipour, 2023). A recent benchmark study evaluated these tools and found their performance to be unsatisfactory, reporting phoneme error rates (PER) between 15-50%, homograph disambiguation accuracy below random baseline, and Ezafe detection F1 scores ranging from 6-60% (Qharabagh et al., 2025b). Subsequently, a Persian LLM-powered G2P model was introduced that substantially improved these metrics (Qharabagh et al., 2025b). Nevertheless, such models are not suitable for free, offline, or real-time use, the key constraints of our target applications.

Building on those findings, another study leveraged the outputs of the LLM-based system to create a new dataset and train two open-source, offline G2P models (Qharabagh et al., 2025a): Homo-GE2PE (Fetrat, 2025a) and HomoFast eSpeak (Fetrat, 2025b). Homo-GE2PE is a high-quality neural G2P model that performs well across PER, homograph disambiguation, and Ezafe detection. HomoFast eSpeak, in contrast, is entirely non-neural and extremely fast, achieving good PER and homograph accuracy but offering limited Ezafe detection capability due to its lack of linguistic modeling.

2.3 Decoupled TTSs

To the best of our knowledge, no prior work has proposed structuring a TTS system so that some of its internal submodules operate as independent services to take advantage of modular decoupling. While several studies and open-source projects provide complete TTS systems as API-based services (Black et al., 2004; MARYTTS, 2022; Pipecat AI, 2024; LlamaEdge contributors, 2024), this should not be confused with the approach presented here.

Our work differs fundamentally from these systems: rather than exposing the entire synthesizer as a remote service, we implement a service-based decomposition within the TTS pipeline itself. This design decouples computationally heavy, higher-latency modules, such as context-aware phonemization components, from the lightweight inference core, improving overall responsiveness and enabling real-time performance.

3 Methodology

As discussed earlier, our baseline system is Piper, which adopts a two-stage pipeline (Type 3 granularity in Figure 2): (1) a text-to-phoneme conversion step implemented with the eSpeak phonemizer, followed by (2) a neural phoneme-to-speech (P2S) model that synthesizes the waveform without a separate vocoder. The primary focus of this work is to strengthen the first stage by introducing context-aware phonemization and to address the practical challenges that arise when integrating this improved G2P component into the complete TTS pipeline.

The default phonemizer in PiperTTS, eSpeak, is a rule-based system relying on dictionary lookups and hardcoded linguistic rules. This design introduces weaknesses for languages that require context-aware phonemization, particularly in handling homograph disambiguation and Ezafe detection in Persian. We propose two complementary families of solutions to address these challenges.

3.1 Statistical Context-Awareness

Context-awareness can be introduced in lightweight TTS systems using simple statistical methods. Certain phonemization tasks, such as homograph disambiguation, can be addressed with shallow contextual statistics instead of heavy neural models.

Qharabagh et al. (2025a) showed that a method based on word co-occurrence distributions can improve homograph disambiguation accuracy by up to 30 percentage points. Their approach constructs a database of homographs and their commonly associated context words, selecting the pronunciation with the highest contextual overlap for a given input. This database includes about 327 thousand balanced samples for 285 homogarphs with an average of 9.4 context words. On average, there are over 1400 unique words in the context sentences of each homograph word in the database. We adopt this lightweight strategy to enhance PiperTTS's phonemizer without adding computational overhead or latency.

3.2 Distilled Linguistic Knowledge

Certain aspects of phonemization require deeper linguistic understanding, such as detecting the Ezafe phoneme, which depends on grammatical and semantic relations between words. However, full-scale language understanding is not necessary for this task. Task-specific, lightweight neural models can be effectively trained via knowledge distillation from larger models.

In our case, Ezafe detection can be viewed as a subtask of part-of-speech (POS) tagging. The SpaCy POS tagger for Persian (Roshan, 2023) is reported to achieve top performance on Ezafe tagging but is relatively heavy and slow during inference (Section 4.2). To obtain a lighter alternative, we distilled the Ezafe tagging knowledge of the SpaCy model into a smaller model based on ALBERT (Lan et al., 2019).

We created a labeled dataset from the text portion of the ManaTTS corpus (Qharabagh et al., 2025c) by automatically annotating Ezafe tags using the SpaCy tagger's predictions. A pretrained Persian ALBERT model[1] (HooshvareLab, 2021) was then fine-tuned on this data, producing a smaller, faster model with performance nearly comparable to the original tagger (Section 4.2). For efficient CPU-based inference and reduced memory usage, the distilled model was exported to ONNX.

[1]The specific version used was HooshvareLab/albert-fa-zwnj-base-v2.

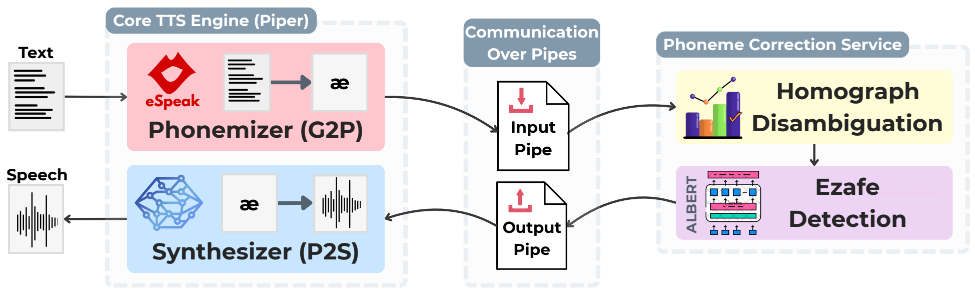

Figure 3: The proposed service-based architecture for context-aware TTS.

3.3 Fine-tuned Synthesizer

We fine-tuned the phoneme-to-speech model on the phoneme sequences produced by the enhanced phonemizer. Fine-tuning was carried out for 1,000 epochs with a batch size of 32 on a workstation equipped with an NVIDIA A100-SXM4 GPU (80 GB) and an Intel Xeon Platinum 8380 CPU with 1 TB of system memory, using the Persian ManaTTS dataset (Qharabagh et al., 2025c). This step was crucial for enabling the model to correctly handle Ezafe phonemes and to distinguish between homographs that differ by only a few phonemes, rather than biasing toward the most frequent pronunciations observed in baseline models. The fine-tuned synthesizer was exported to ONNX for faster and more lightweight CPU inference.

3.4 Service-Based Integration

When all components of a TTS system are integrated into a single unified runtime, the individual loading and inference delays of each module accumulate, resulting in a significant overall latency. To overcome this bottleneck, we adopted a service-oriented architecture, setting up utility modules as independent, persistent services running in separate processes. The core TTS module communicates with these services using inter-process communication (IPC) via piped input and output files. This design decouples the initialization of independent modules and significantly reduces latency during inference.

In our setup, the context-aware phonemization components operate as a dedicated service, with the core TTS engine interacting through two file pipes (input and output). For each input text, the core TTS module first generates an initial phoneme sequence using its default phonemizer (PiperTTS's eSpeak-based component). This sequence is then sent to the context-aware phonemization service, where it undergoes two refinement stages: the homograph disambiguation module corrects potential mispronunciations, and the Ezafe detection model inserts any missing Ezafe phonemes. The enhanced phoneme sequence is then returned to the core TTS engine and passed to the phoneme-to-speech model, which synthesizes the final audio output. Figure 3 illustrates this service-based setup for the context-aware TTS proposed in this study.

4 Experiments

In this section, we evaluate how the proposed context-aware phonemization modules improve a rule-based phonemizer's accuracy and affect inference speed. While context-aware modules enhance phonemization quality, they also introduce additional computational load that can increase latency. We therefore assess their performance within our service-based framework, which decouples these heavier components and restores real-time operation without compromising phonemization improvements.

Our enhanced system, Piper equipped with the proposed lightweight context-aware phonemization components, is referred to as "Piper + LCA-G2P", where LCA stands for Lightweight Context-Aware. To demonstrate the framework's ability to handle heavier models, we also integrated the state-of-the-art Persian G2P model, Homo-GE2PE (Qharabagh et al., 2025a), which handles both homograph disambiguation and Ezafe detection. This setup, "Piper + Neural G2P", uses a substantially larger model (300M parameters) compared to Piper's lightweight 15-20M parameters, showing that the framework can accommodate computationally intensive neural components while maintaining real-time performance.

All experiments evaluating real-time factor (RTF)[2] were conducted on a typical end-device configuration: a Windows system with a 12th Gen

[2]RTF, or real-time factor, is the ratio of audio synthesis time to the duration of the generated audio. For instance, an RTF of 0.2 indicates synthesis five times faster than real-time.

Model	**PER** (% ↓)	**Ezafe F1** (% ↑)	**Homograph** Acc. (% ↑)	**RTF ↓** Direct Call	Service-Based
MatchaTTS (Mahmoudi, 2025)	6.32 ± 0.00	19.58 ± 0.00	43.87 ± 0.00	0.185 ± 0.051	–
GlowTTS (Kamtera, 2023)	6.61 ± 0.00	19.96 ± 0.00	43.87 ± 0.00	1.364 ± 0.705	–
Piper (Base) (Karimi, 2024)	6.32 ± 0.00	19.58 ± 0.00	43.87 ± 0.00	**0.153 ± 0.012**	–
Piper + Neural G2P	<u>4.95 ± 0.68</u>	<u>87.70 ± 0.78</u>	<u>74.53 ± 0.39</u>	3.840 ± 0.415	0.396 ± 0.095
Piper + LCA G2P	**4.80 ± 1.06**	**90.08 ± 0.72**	**77.67 ± 0.22**	5.519 ± 0.984	<u>0.167 ± 0.015</u>

Table 1: Comparison of phonemization accuracy and inference speed across baseline and proposed TTS models.

Model	**Params** (Millions ↓)	**Memory** (MB ↓)	**Disk** (MB ↓)	**Ezafe F1** (% ↑)	**Avg. Inf. Time** (s ↓)
SpaCy (Roshan, 2023)	162.84	621.19	1258.49	**97.67 ± 0.00**	0.110 ± 0.004
ALBERT-based (Ours)	**11.09**	**42.32**	**41.38**	94.19 ± 0.00	**0.037 ± 0.001**

Table 2: Comparison between the SpaCy teacher model and the distilled ALBERT-based Ezafe detector.

Intel Core i7-1255U CPU (10 cores, 1.7 GHz) and 16 GB of RAM, running CPU-only inference. This setup demonstrates the system's suitability for offline, low-latency, and real-time applications, without relying on GPU acceleration. The results are summarized in Table 1.

4.1 Mean Opinion Score

The enhanced soundness and context-awareness of the phonemization process, along with the subsequent fine-tuning of the phoneme-to-speech (P2S) engine, are expected to improve the overall naturalness of the generated speech. Table 3 presents the Mean Opinion Score (MOS) results for the baseline and enhanced TTS systems, as well as the reference natural speech.

Table 3 shows the average Mean Opinion Score (MOS) for the baseline and enhanced TTS systems, alongside natural speech, based on evaluations from 16 native Persian speakers across seven utterances. For full details of the experiment, please refer to Appendix B.

4.2 Ezafe Detection Module Evaluation

This section presents the experiments conducted on the distilled Ezafe detection module, demonstrating that it achieves a substantial reduction in size and computational overhead while retaining the strong performance of its teacher model. All experiments were conducted on a CPU environment in Google Colab. The results are summarized in Table 2.

Source	**MOS ↑**
Glow (Kamtera, 2023)	1.30 ± 0.75
Matcha (Mahmoudi, 2025)	2.54 ± 0.99
Piper (Base) (Karimi, 2024)	2.41 ± 0.84
Piper + LCA G2P (Ours)	**3.14 ± 1.00**
Natural Speech	4.21 ± 0.97

Table 3: MOS of the baseline and enhanced TTS system compared to natural speech.

5 Conclusion

This study addressed the fundamental trade-off between speed, lightweightness, and context-aware phonemization in G2P-aided TTS systems. We proposed practical approaches to mitigate this challenge, including methods for developing auxiliary context-aware modules that are inherently lighter and faster, as well as introducing a service-based architecture that enables their efficient integration into real-time TTS pipelines.

The proposed framework demonstrated that it is possible to achieve enhanced phonemization accuracy without compromising real-time performance. By decoupling heavy context-aware components from the core runtime and executing them as independent services, the system maintained low-latency while significantly improving the overall soundness of the synthesized speech. These characteristics make the architecture particularly suitable for offline, end-device, and low-latency applications such as screen readers.

All source code, models, and experimental results from this work are publicly available. [3]

Limitations

Even with fully corrected phoneme sequences in the TTS system, achieving complete naturalness remains out of reach. This is primarily because lightweight TTS models have limited capacity in the phoneme-to-speech component, which is typically insufficient to fully capture or reproduce higher-level prosodic and expressive features. As a result, the overall perceived naturalness cannot reach its maximum potential. Further research is needed to improve these aspects of naturalness while maintaining the desired properties of speed and lightweight design.

Another consideration is that, from a perceptual standpoint, naturalness is more closely associated with qualities such as smoothness, noiselessness, and accurate intonation and stress patterns. Correct phonemization primarily affects pronunciation soundness and only indirectly contributes to perceived naturalness. It may therefore be valuable to design subjective evaluation protocols that separate the assessment of phonemization accuracy from other dimensions of naturalness, such as prosody and fluency.

Another limitation, or rather an avenue for future enhancement, lies in the service-based setup itself. Now that several components are decoupled from the core TTS engine, additional optimization strategies can be applied to the service layer. For example, implementing request-level parallelism or asynchronous processing could further reduce overall system latency and improve scalability.

Acknowledgments

This research was supported by the IR National Science Foundation (INSF) Grant No. 4033002.

[3] `https://github.com/MahtaFetrat/Piper-with-LCA-Phonemizer`

References

Mohammad Hasan Sohan Ajini. 2022. Attention based grapheme to phoneme. Accessed: 2025-11-17.

Sajad Alipour. 2023. Persian grapheme to phoneme with transformer. Accessed: 2025-11-17.

Kenneth R Beesley and Lauri Karttunen. 2003. Finite-state morphology: Xerox tools and techniques. *CSLI, Stanford*, pages 359–375.

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

Alan W. Black, Paul Taylor, and 1 others. 2004. *Festival Speech Synthesis System: Server/Client API*. Section 28.3: BSD socket server; long-lived synthesis process.

Alan W Black and Paul A Taylor. 1997. Automatically clustering similar units for unit selection in speech synthesis.

Hafez Dehghani. 2022. persian_phonemizer: A tool for translating persian text to ipa. Accessed: 2025-11-17.

Homer Dudley, Richard R Riesz, and Stanley SA Watkins. 1939. A synthetic speaker. *Journal of the Franklin Institute*, 227(6):739–764.

eSpeak NG developers. 2013. espeak NG text-to-speech engine. GitHub repository. Accessed: 2025-11-17.

Mahta Fetrat. 2025a. Homo-ge2pe-persian: Persian grapheme-to-phoneme conversion with homograph disambiguation. Accessed: 2025-11-08.

Mahta Fetrat. 2025b. Homofast-espeak-persian: A homograph-aware persian g2p extension of espeak ng. Accessed: 2025-11-08.

HooshvareLab. 2021. Albert-persian: A lite bert for self-supervised learning of language representations for the persian language. Model: albert-fa-zwnj-base-v2. Accessed: 2025-11-07.

Andrew J Hunt and Alan W Black. 1996. Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE international conference on acoustics, speech, and signal processing conference proceedings*, volume 1, pages 373–376. IEEE.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379.

Kamtera. 2023. persian-tts-female-glow_tts: Single-speaker female persian text-to-speech model. Accessed: 2025-11-07.

Sadegh Karimi. 2024. persian-text-to-speech: Persian text-to-speech model. Accessed: 2025-11-17.

Heeseung Kim, Sungwon Kim, and Sungroh Yoon. 2022. Guided-tts: A diffusion model for text-to-speech via classifier guidance. In *International Conference on Machine Learning*, pages 11119–11133. PMLR.

Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon. 2020. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33:8067–8077.

Jaehyeon Kim, Jungil Kong, and Juhee Son. 2021. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning*, pages 5530–5540. PMLR.

Dennis H Klatt. 1980. Software for a cascade/parallel formant synthesizer. *the Journal of the Acoustical Society of America*, 67(3):971–995.

Dennis H Klatt. 1987. Review of text-to-speech conversion for english. *The Journal of the Acoustical Society of America*, 82(3):737–793.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, and 1 others. 2023. Voicebox: Text-guided multilingual universal speech generation at scale. *Advances in neural information processing systems*, 36:14005–14034.

Yinghao Aaron Li, Cong Han, and Nima Mesgarani. 2025. Styletts: A style-based generative model for natural and diverse text-to-speech synthesis. *IEEE Journal of Selected Topics in Signal Processing*.

Yinghao Aaron Li, Cong Han, Vinay Raghavan, Gavin Mischler, and Nima Mesgarani. 2023. Styletts 2: Towards human-level text-to-speech through style diffusion and adversarial training with large speech language models. *Advances in Neural Information Processing Systems*, 36:19594–19621.

LlamaEdge contributors. 2024. Tts-api-server: Restful api server for piper (openai-compatible routes). GitHub repository. Accessed: 2025-11-17.

Ali Mahmoudi. 2025. Khadijah-fa_en-matcha-tts-model: A persian/english text-to-speech model using matcha-tts. Accessed: 2025-11-17.

MARYTTS. 2022. Mary tts: an open-source, multilingual text-to-speech synthesis system. Accessed: 2025-11-16.

Shivam Mehta, Ruibo Tu, Jonas Beskow, Éva Székely, and Gustav Eje Henter. 2024. Matcha-tts: A fast tts architecture with conditional flow matching. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11341–11345. IEEE.

David R Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision g2p for many languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Amir Hossein Navabi. 2025. persian-tts-piper: A hugging face space for persian text-to-speech using piper. Accessed: 2025-11-07.

OHF-Voice. 2025. piper1-gpl: Fast and local neural text-to-speech engine. Accessed: 2025-11-07.

Musharraf Omer. 2025. sonata-nvda: A speech synthesizer driver for nvda using neural tts models. Accessed: 2025-11-07.

Kyubyong Park. 2019. g2pE: A simple english grapheme-to-phoneme converter (seq2seq/transformer implementation). GitHub repository.

Demetry Pascal. 2020. Simple persian (farsi) grapheme-to-phoneme converter. Accessed: 2025-11-17.

Pipecat AI. 2024. Piperttsservice: Self-hosted http server for piper tts. Product documentation. Accessed: 2025-11-17.

Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. 2021. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International conference on machine learning*, pages 8599–8608. PMLR.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. 2019. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE.

Mahta Fetrat Qharabagh, Zahra Dehghanian, and Hamid R Rabiee. 2025a. Fast, not fancy: Rethinking g2p with rich data and rule-based models. *arXiv preprint arXiv:2505.12973*.

Mahta Fetrat Qharabagh, Zahra Dehghanian, and Hamid R Rabiee. 2025b. Llm-powered grapheme-to-phoneme conversion: Benchmark and case study. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Mahta Fetrat Qharabagh, Zahra Dehghanian, and Hamid R Rabiee. 2025c. Manatts persian: a recipe for creating tts datasets for lower resource languages. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9177–9206.

Azam Rabiee. 2019. Persian g2p. GitHub repository. Accessed: 2025-11-17.

Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2020. Fastspeech 2: Fast and high-quality end-to-end text to speech. *arXiv preprint arXiv:2006.04558*.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. 2019. Fastspeech: Fast, robust and controllable text to speech. *Advances in neural information processing systems*, 32.

Rhasspy Team. 2023. Piper: Fast, local neural text to speech. GitHub repository.

Roshan. 2023. spacy_pos_tagger_parsbertpostagger. Accessed: 2025-11-13.

Yoshinori Sagisaka. 1988. Speech synthesis by rule using an optimal selection of non-uniform synthesis units. In *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, pages 679–682. IEEE.

Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, and 1 others. 2018. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4779–4783. IEEE.

Keiichi Tokuda, Takayoshi Yoshimura, Takashi Masuko, Takao Kobayashi, and Tadashi Kitamura. 2000. Speech parameter generation algorithms for hmm-based speech synthesis. In *2000 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 00CH37100)*, volume 3, pages 1315–1318. IEEE.

Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, and 1 others. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12:1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, and 1 others. 2023. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*.

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, and 1 others. 2017. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*.

Ryuichi Yamamoto, Eunwoo Song, and Jae-Min Kim. 2020. Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6199–6203. IEEE.

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

ZachB100. 2023. Piper training guide with screen reader. GitHub repository. Accessed: 2025-11-17.

Heiga Zen, Keiichi Tokuda, and Alan W Black. 2009. Statistical parametric speech synthesis. *speech communication*, 51(11):1039–1064.

A Selection Criteria for PiperTTS

Considering the requirements of the TTS system in this study and the models reviewed in the related work section, we found PiperTTS to be the most suitable architecture for our needs. PiperTTS has been available for several years and has gained significant community attention and contributions (OHF-Voice, 2025). It exhibits several characteristics that align with the requirements and objectives of this research:

- **Lightweightness:** One of PiperTTS's core strengths is its ability to run efficiently on CPUs and even low-end devices such as Raspberry Pi. Its ONNX runtime export enables lightweight deployment on various hardware platforms.
- **Speed:** PiperTTS demonstrates high inference speed, achieving a real-time factor (RTF) of approximately 0.2.
- **Naturalness:** The model provides medium to high perceptual quality, as verified through publicly available checkpoints (Navabi, 2025).
- **Accessibility Integration:** PiperTTS has already been integrated into the open-source NVDA screen reader, which embeds TTS engines through add-ons called synthesizer drivers. An established Piper synthesizer driver is publicly available (Omer, 2025).
- **Open Source:** Being open source, PiperTTS facilitates the development of accessible tools and enables contributions to a field that currently lacks substantial open research.

- **Persian Support:** The model has an active Persian-speaking community, with available checkpoints and established Persian training setups.

Given these factors, PiperTTS was selected as the base TTS architecture for this work.

B Mean Opinion Score Details

To evaluate perceived naturalness, we selected seven utterances from a recent issue of the online monthly magazine *Nasl-e-Mana*, a publication for the blind community and the source of the publicly available ManaTTS dataset. The chosen issue contained content not included in the ManaTTS corpus used for fine-tuning the phoneme-to-speech model.

For each utterance, audio was generated using five sources: two open-source lightweight Persian TTS models (GlowTTS and MatchaTTS), the baseline PiperTTS model, our enhanced Piper system, and the corresponding natural speech recordings. The audio samples for all utterances are provided in the repository's samples directory. [4] To avoid bias based on overall model reputation or perceived quality, the order of the five sources was independently shuffled for each utterance.

Sixteen native Persian speakers were asked to rate the naturalness of each audio clip on a scale from 1 to 5 (MOS), with 5 indicating the most natural pronunciation. Participants were instructed as follows (translated from Persian):

"Please listen to each audio clip and rate its naturalness on a scale from 1 to 5. A score of 5 corresponds to fully natural pronunciation, while a score of 1 corresponds to the least natural or highly robotic pronunciation. Lower your rating if you detect any unnatural intonation, mispronunciation, or mechanical quality."

The resulting overall MOS values, averaged across all utterances, are shown in Figure 4. Detailed MOS results per utterance, including the shuffled order of sources, are provided in Table 4. Standard deviations are reported to reflect inter-subject variability. Additionally, the distribution of MOS scores assigned to our enhanced model by individual participants is illustrated in Figure 5.

Additional Figures

Visualizing experimental findings can provide valuable insight. A central concern of this study is the trade-off between inference speed and phonemization quality: as the quality of TTS improves, especially with the aid of context-aware and linguistically-informed phonemization tools, inference typically becomes slower. Our work proposes a service-based architecture that mitigates this trade-off, allowing models to achieve both reasonable speed and improved phonemization quality.

Figure 6 illustrates the performance of the evaluated models along two axes: speed and quality. Inference speed is represented by the real-time factor (RTF), displayed on a logarithmic scale. For quality, we define a composite metric that combines the key context-aware phonemization challenges studied in this work:

$$\text{G2P Quality} = \frac{\text{Ezafe F1} + \text{Homograph Acc.}}{\text{PER}} \tag{1}$$

This formulation rewards higher Ezafe F1 and homograph accuracy while penalizing higher phoneme error rate, providing an intuitive measure aligned with our goals. The quality values are schematically arranged for visualization, maintaining relative correctness.

In this plot, models that are both fast and high-quality appear in the top-right corner. As expected, our proposed enhanced version, "Piper + LCA-G2P", occupies this position, demonstrating that it maintains strong inference speed while substantially improving phonemization quality.

[4] https://github.com/MahtaFetrat/Piper-with-LCA-Phonemizer

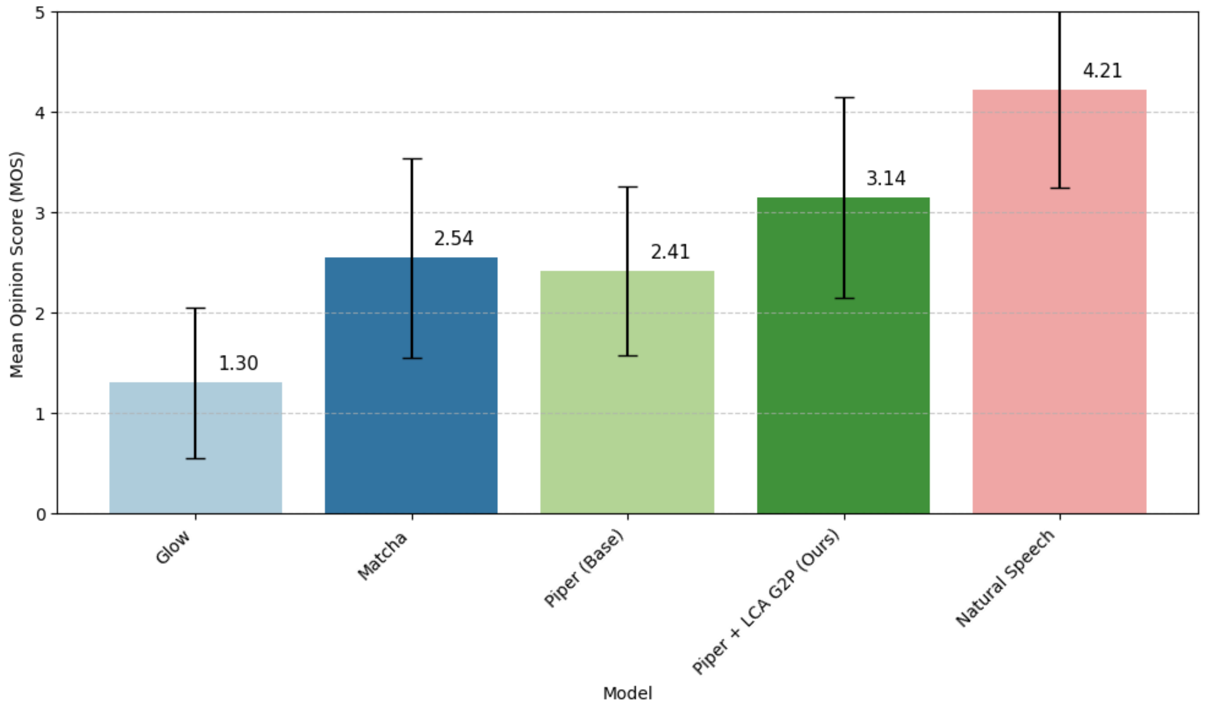

Figure 4: Overall average MOS across all seven utterances for each TTS system and natural speech, highlighting the improved naturalness of the enhanced Piper system.

		Piper + LCA	**Natural**	**Glow**	**Matcha**	**Piper (Base)**
Utterance 1	**MOS**	2.94 ± 0.68	4.12 ± 0.50	1.19 ± 0.54	2.25 ± 0.77	2.38 ± 0.81
	Order	5	1	2	3	4
Utterance 2	**MOS**	3.75 ± 0.93	4.25 ± 1.00	2.00 ± 1.03	2.62 ± 1.09	2.38 ± 0.89
	Order	3	2	4	1	5
Utterance 3	**MOS**	3.19 ± 0.91	4.88 ± 0.34	1.12 ± 0.50	2.44 ± 0.96	2.12 ± 0.62
	Order	4	3	2	5	1
Utterance 4	**MOS**	2.81 ± 0.98	4.56 ± 1.03	1.12 ± 0.34	2.50 ± 1.03	2.19 ± 1.05
	Order	2	5	4	3	1
Utterance 5	**MOS**	2.62 ± 1.15	4.31 ± 0.70	1.19 ± 0.75	2.69 ± 1.08	3.00 ± 0.73
	Order	2	4	1	5	3
Utterance 6	**MOS**	3.69 ± 0.87	3.81 ± 1.22	1.25 ± 0.77	2.62 ± 1.02	2.62 ± 0.81
	Order	2	1	5	3	4
Utterance 7	**MOS**	3.00 ± 1.03	3.56 ± 1.15	1.25 ± 0.77	2.62 ± 1.09	2.19 ± 0.75
	Order	1	2	4	5	3

Table 4: Per-utterance MOS (mean $\pm$ std) for each source. "Order" shows the presentation order (1–5) of the sources for that utterance (randomized per utterance).

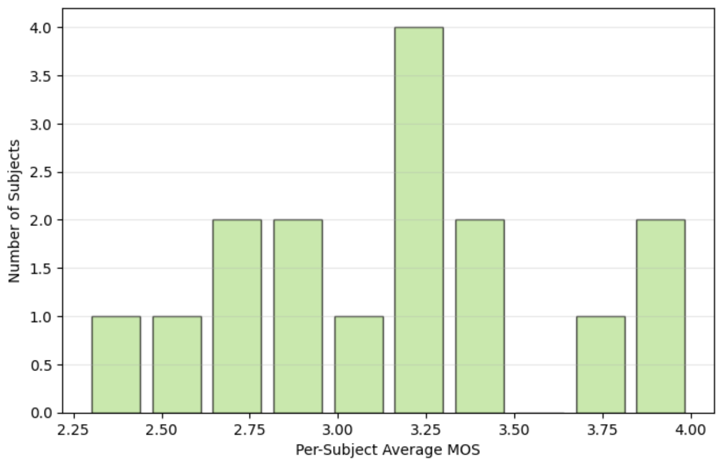

Figure 5: Distribution of MOS ratings assigned by participants to the enhanced TTS system (Piper + LCA-G2P).

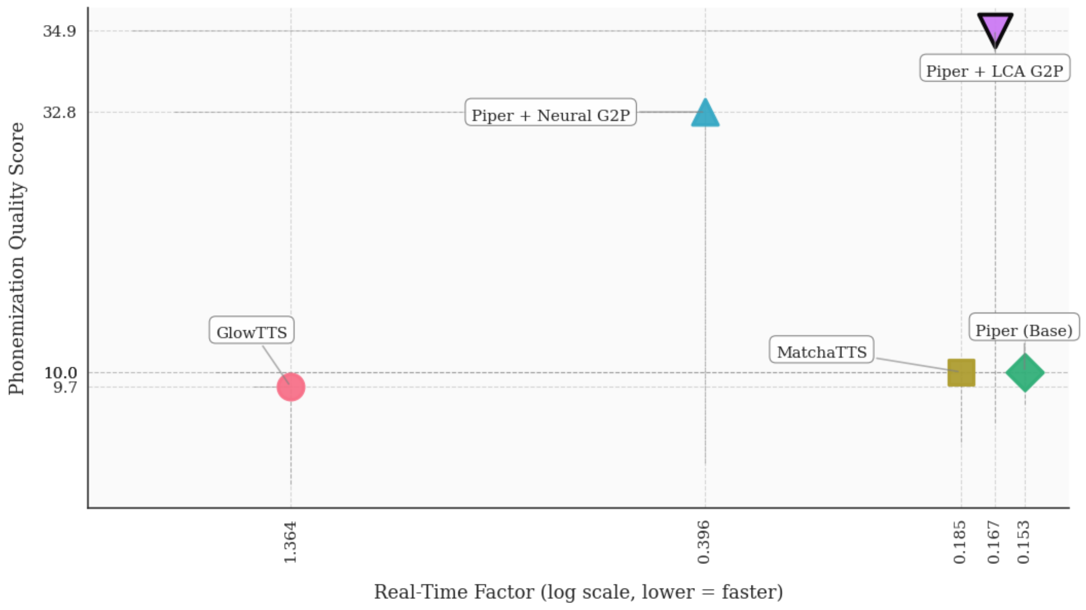

Figure 6: Trade-off between inference speed (RTF, log scale) and phonemization quality (composite metric) for various TTS models. The top-right region indicates models that are both fast and high-quality.