

Teaching a Lightweight Neural TTS Engine to Read: Span-Preserving Text Preparation, Phoneme-Set-Aware Pronunciation Repair and Model-Timed Subtitles for Piper

Asad Shehzad
Loudness.ai

Technical report LA-RB-2026-002, version 1.0.1, 15 October 2026
<https://loudness.ai/research/piper-tts-text-normalization/>

Abstract

Piper is a widely used neural text-to-speech engine that runs in real time on ordinary processors. It reads text through the *espeak-ng* phonemizer and performs no text normalization of its own, so written forms such as prices, dates, abbreviations, units and phone numbers are often read literally. We describe a text preparation layer for Piper that rewrites such forms while keeping, for every spoken word, the span of the written text it came from; it also segments text into sentences, lines, paragraphs and clauses with explicit pauses. We report two further observations: several community English (US) Piper voices were trained on *espeak*’s British phoneme set and therefore pronounce words such as *schedule* and *privacy* the British way, which we repair with phoneme strings restricted to the symbols those voices were trained on; and Piper’s duration output yields exact word timings for subtitles once spoken words are mapped back to written words with a small dynamic programme. On a suite of 84 hand-written English and Spanish test sentences the phoneme error rate against reference readings fell from 32.6% to 0.9%, false sentence endings fell from 18 to 0, and none of 19 control sentences changed. On a held-out half of NVIDIA NeMo’s public text normalization test cases, which we did not write, the phoneme error rate was 41.0% for unprocessed text, 28.5% for a typical number-to-words script and 15.8% for our layer in English (200 cases), and 64.7%, 56.2% and 43.2% in Spanish (277 cases). We release the code and evaluation data under the Apache License 2.0.

1 Introduction

Lightweight neural text-to-speech (TTS) engines make speech synthesis practical on ordinary servers and devices. Piper [1], built on VITS [3], is a common choice: its voices are small, it runs faster than real time on a few processor cores, and more than a hundred voices exist in dozens of languages. Piper converts text to phonemes with *espeak-ng* [2] and synthesizes speech sentence by sentence. It does not normalize text, and *espeak-ng*’s own rules are tuned for plain words and numbers. In practice, users who deploy Piper write their own pre-processing scripts, typically around a number-to-words library.

Text normalization for TTS is a mature research area, from the taxonomy of non-standard words of Sproat et al. [4] and production weighted finite-state systems [5, 8] to neural approaches [6, 7]. Our aim is narrower and practical: to describe what a small, dependency-free preparation layer must do for Piper specifically, to measure how much it helps against both unprocessed text and a typical community script, and to report two Piper-specific issues that we did not find documented.

Our contributions are:

1. A measurement of how `espeak-ng`, as used by Piper, reads everyday written forms in English and Spanish.
2. A span-preserving rule layer in which each rule claims a span of the original text, so rules cannot rewrite each other’s output and every spoken word can be traced to the written word it came from; with segmentation into pause classes and click-free joins.
3. The observation that some community English (US) Piper voices were trained on `espeak`’s British phoneme set, and a repair that supplies American pronunciations using only symbols in that set.
4. Word timings from Piper’s duration output, mapped to written words with a dynamic programme, for subtitles and word-level highlighting; and the observation that loading the duration-exposing model from memory roughly doubles memory use compared with loading it from a file.
5. An evaluation on our own suite, on an independent held-out set, an ablation of rule groups, and a timing consistency check, with code and data released.

2 Background

Piper and `espeak-ng`. Piper voices are VITS models trained on phoneme sequences produced by a particular `espeak-ng` voice, which each model’s configuration records (for example `en-us` or `en`). At inference, Piper phonemizes text with the same `espeak-ng` voice, splits it into sentences at `espeak-ng`’s clause boundaries, maps phoneme symbols to identifiers and synthesizes each sentence. Inference parameters control speaking rate (*length scale*) and the random variation of sound and timing (*noise scale* and *noise width*). Piper accepts raw phonemes between double square brackets. Version 1.8 can expose the per-phoneme durations predicted by the model by marking the duration tensor as an additional output.

Text normalization. Normalization converts non-standard words—numbers, dates, currency, abbreviations, addresses—into the words a reader would say [4]. Production systems use language-specific grammars compiled to weighted finite-state transducers [5, 8]; neural systems learn the mapping but can produce unrecoverable errors [6, 7]. NVIDIA NeMo text processing publishes grammars and test cases for many languages [9]; we use its English and Spanish test cases as independent evaluation data.

Timing. Word timings for synthesized speech are usually obtained after synthesis by forced alignment [10] or speech recognition with timestamps [11]. A non-autoregressive model such as VITS already predicts how long each phoneme lasts, so timings can be read from the model itself.

3 Method

3.1 Span-preserving rules

Each line of text is processed by an ordered list of rules. A rule is a regular expression applied to the *original* line together with a function that returns what to read instead (or declines). The first

rule to claim a span wins; later rules cannot match inside a claimed span. The line is then rebuilt from unchanged text and replacements, and every character of the result carries the span of the original it came from. Two properties follow: rules cannot corrupt one another (an e-mail address claimed by the address rule is never touched by the abbreviation or slash rules), and each word of the prepared text maps to a span of the written text, so that *dollars* in “5 dollars and 50 cents” points back to “\$5.50”. A rule that keeps its first word unchanged (“Felipe VI” → “Felipe sexto”) claims only the remainder, so each written word keeps its own spoken words. User-supplied double square brackets are neutralized so that raw phonemes cannot be injected.

3.2 Coverage

English and Spanish have full rule sets: titles and street words (keeping a sentence end when an abbreviation closes a sentence, as in “Apple Inc. We...”), money with singular and plural forms and regional currencies (“\$” is read as pesos for Mexico), numeric, ISO and dotted dates in each region’s order with American year pairs (“twenty twenty-six”) for US English, clock times including seconds, decades, units, number ranges, phone numbers in digit groups (including Spain’s nine-digit format), fractions, Roman numerals after keywords and regnal names, initials, thousands grouped with spaces and slashes between words. All languages receive the layout rules: list markers, markdown and emoji are removed, web addresses are shortened by dropping the scheme and “www”, and lines without final punctuation receive a full stop so that they end with a falling tone.

3.3 Segmentation and pauses

The prepared text is split into segments, each followed by a pause class: sentence, list line, paragraph, ellipsis or clause. Sentence boundaries are taken only at a full stop, question or exclamation mark not followed by a lowercase letter, after abbreviations have been expanded. Unpunctuated runs longer than 28 words are split before a conjunction near the middle where one exists (seven languages have conjunction lists), otherwise at a word gap, and the first part ends with a comma so that the voice keeps a continuing tone. With p the user’s chosen pause, gaps are p after a sentence, line or ellipsis, $0.4p$ after a clause and $\max(2p, p + 0.3\text{ s})$ after a paragraph. Each segment is synthesized separately and joined with exact silences. Because the voices often end on a non-zero sample (28–57 dB below the peak in our measurements), a 3 ms fade-in and 12 ms fade-out are applied at every join, and a 120 ms tail is appended.

3.4 Pronunciation repair for voices trained on British phonemes

Several community English (US) voices, trained from scratch on American audiobook speakers (LibriTTS-R [12] derived data), record *espeak*’s **en** voice, which is British English. For most words this is harmless: the model learned to realize British symbols with the American sounds of its speakers. It matters where the British form is a different *lexical* pronunciation rather than a different accent: *espeak en* gives $\text{f}^{\text{h}}\text{edju:l}$ for *schedule*, $\text{t}\text{əm}^{\text{h}}\text{ɑ:t}\text{əu}$ for *tomato*, $\text{p}\text{r}\text{ɪ}^{\text{h}}\text{vəsi}$ for *privacy* and $\text{z}^{\text{h}}\text{ed}$ for the letter Z, and the voices say exactly that, whereas **en-us** gives $\text{sk}^{\text{h}}\text{ed}^{\text{h}}\text{z}\text{u:l}$, $\text{t}\text{əm}^{\text{h}}\text{e}\text{r}\text{əu}$, $\text{p}\text{r}\text{ɪ}^{\text{h}}\text{əvəsi}$ and $\text{z}^{\text{h}}\text{ɪ}$. Switching these voices to **en-us** phonemes is not a remedy, because their training data never contained symbols such as æ or ɹ . Instead we wrote American forms of 118 frequent words in the British symbol set—including *-ary* and *-ory* words, yod-dropping words [13] such as *new* and *student*, and the letter Z—and pass them as raw phoneme blocks only to US voices whose model records **en**. A unit test checks that no block uses a symbol outside the set. Two Piper details matter: punctuation directly after a raw block is dropped and the next word is joined to the block,

so punctuation must be placed inside the block; and a raw block followed by text leaves a doubled word gap, which we collapse before synthesis.

3.5 Word timings from the model’s durations

For each segment we obtain, from Piper’s duration output, the start and end sample of every phoneme symbol, including punctuation symbols, whose durations carry the pauses. Spoken words are aligned to this stream as follows. Let $c_1, \dots, c_T$ be the number of symbols of each prepared word when phonemized on its own, rescaled so that $\sum c_k = N$, the number of non-space symbols in the stream, and let G be the set of positions where a spoken word begins. We choose cut points $0 = b_0 < b_1 < \dots < b_T = N$ minimizing

$$\sum_{k=1}^T |(b_k - b_{k-1}) - c_k| + \lambda \sum_{k=1}^{T-1} [b_k \notin G], \quad (1)$$

with $\lambda = 1.5$ and each b_k searched within eight symbols of its proportional position, by dynamic programming. This tolerates *espeak-ng* merging short words (“for the” became one spoken word in our tests) and expanding one written word into several (“2026”). Each prepared word receives the time of its symbols; written words receive the union of the times of the prepared words that point to them, and a group of prepared words covering several written words (“5 euros con 50 céntimos” for “5,50 €”) shares its time in proportion to the written words’ lengths. Subtitle cues are built from segments: one cue per sentence or line, longer ones split at word gaps to fit two lines of about 42 characters, and a cue ending mid-thought (comma, ellipsis, colon) is joined to the next when both fit.

Loading the timing-capable model. Exposing the duration tensor requires editing the model graph. Piper’s loader can do this in memory with the `onnx` package. In our deployment this used 811 MB for four voices against 480 MB when the same voices were loaded from files; writing each edited model to disk once and loading it from the file returned memory use to 486 MB. Edited copies of the models behind the 33 voices exercised in our release check took 2.1 GB of disk.

4 Experimental setup

Software. `piper-tts` 1.8.0 with its bundled *espeak-ng* 1.52.0.1, `onnxruntime` 1.19.2, NumPy 2.0.2 and Python 3.9.6 on an Apple M4 laptop running macOS 26.6.1 for all text measurements; `onnxruntime` 1.31.0, NumPy 2.5.4 and Python 3.12.3 on a 12-vCPU AMD EPYC virtual server for production checks. The baseline script uses `num2words` 0.5.14.

Metric. For each case, the system’s output and a reference reading are phonemized with the same *espeak-ng* voice (**en-us** for US English, **en** for British English, **es** for Spanish). After removing stress marks, punctuation and word gaps, the *phoneme error rate* (PER) is the Levenshtein distance between the two symbol strings divided by the reference length; a case is *exact* when PER is zero. *False sentence endings* count the difference in the number of falling sentence ends (parts ending in a full stop, question or exclamation mark). Because references are themselves read by *espeak-ng*, the metric isolates normalization from *espeak-ng*’s reading of plain words.

Own suite. 103 sentences (71 English, 32 Spanish): 84 test cases in 13 categories with hand-written reference readings, and 19 control sentences of plain text whose reference is the text itself, so that any change to a control is harm. Two known failures (a heteronym and an ambiguous unit) are included deliberately.

Independent set. English and Spanish written-to-spoken test cases from NVIDIA NeMo text processing at commit `ddadfb2` [9] (957 usable cases). The cases were split into two halves by a hash of the written text. After a first run, we studied individual failures in the development half only and corrected general gaps (numbers grouped with spaces, Spanish ordinals written “10.^o”, Spanish prices with the symbol after the number, dotted and year-first dates, lowercase month names, times with seconds, a markdown rule that removed asterisks between digits). A held-out ablation then showed that the number-grouping rule slightly increased English error; we investigated it on development cases only, which revealed that it joined a day and a year (“june 20 2014”) and that lowercase or capitalized month abbreviations were not recognized, and corrected both. We report results on the held-out half, whose individual cases we did not inspect; the first run had printed aggregate scores per category over all cases before the split, and the ablation above was an aggregate held-out signal. After the evaluation, while checking usage guidance, we found that Spanish phone numbers in Spain’s grouped format (“912 345 678”) and amounts whose thousands are grouped with spaces before a currency or unit (“1 500 kg”) were joined into one number; we corrected both, and no reported number changed. NeMo’s references follow NeMo’s reading style, so a correct reading in another style—“two thousand twelve” for NeMo’s “twenty twelve”, or our deliberate omission of “https://www.”—counts as an error for every system.

Systems. *Raw*: the text as Piper receives it. *Script*: a typical community preprocessor that we wrote to be capable and fair, combining num2words with currency, percentages, ordinals, years, decimals, numeric dates, clock times, thousands separators and common abbreviations. *Ours*: the layer described above. NeMo’s own normalizer was not evaluated on its own test cases.

5 Results

Own suite. Table 1 shows that the layer removed almost all differences from the reference readings in the categories it targets and introduced no change to any control sentence. The four remaining differences are the two deliberate failures, a fraction whose Spanish gender the rule cannot know (*media taza*), and a spelling variant in a reference (*Missus* for *Mrs*). Because we wrote both the rules and this suite, these figures show coverage of the intended cases, not generalization.

Independent set. On the held-out half (Table 2) our layer had the lowest error in both languages. Per category (Table 3), the largest gains were for money, telephone numbers, times, dates and fractions. Our layer was worse than raw text for web addresses, because we drop the scheme and “www” by design, and the script was better for Spanish cardinals and times: *espeak-ng* reads large Spanish numbers such as 10^9 as *uno mil millones*, which the script avoids by verbalizing numbers itself, and NeMo reads *01:00* as *una* where we say *una en punto*.

Ablation. Money handling contributed most in English and number grouping most in Spanish (Table 4); a negative value means the group’s removal would lower the error on this set. Groups that target forms absent from NeMo’s cases (titles, layout, pauses) show no effect here and are evaluated by the own suite.

Group	n	PER (%)		Exact		False ends (raw→ours)
		raw	ours	raw	ours	
All test cases	84	32.6	0.9	10	80	18→0
English	58	33.6	0.9	6	55	13→0
Spanish	26	30.4	0.8	4	25	5→0
Controls (unchanged text)	19	0.0	0.0	19	19	0→0
brand name	1	20.8	0.0	0	1	0→0
dates	6	41.8	0.0	0	6	1→0
decades	3	11.5	0.0	1	3	0→0
initials	2	0.0	0.0	2	2	3→0
known limits	2	23.4	23.4	0	0	0→0
layout	7	72.6	0.0	2	7	3→0
money	9	50.5	0.0	0	9	0→0
phone numbers	3	85.2	0.0	0	3	0→0
ranges, fractions, slashes	9	25.5	2.4	0	8	0→0
Roman numerals	5	26.7	0.0	0	5	0→0
times	6	27.7	0.0	0	6	0→0
titles	21	15.3	0.3	5	20	11→0
units	10	31.8	0.0	0	10	0→0

Table 1: Own suite. PER against hand-written reference readings, number of exactly correct cases and false sentence endings, without (raw) and with (ours) text preparation.

Half	Language	n	PER (%)			Exact cases	
			raw	script	ours	raw	ours
held-out	English	200	41.0	28.5	15.8	65	99
held-out	Spanish	277	64.7	56.2	43.2	47	64
development	English	221	41.9	29.4	19.5	68	106
development	Spanish	259	64.8	55.4	36.9	57	84

Table 2: NeMo test cases. Mean PER for raw text, the number-to-words script and our layer, and exact cases. The held-out half was not inspected; the development half is shown for transparency.

Timing. Across 414 synthesized takes of the English test sentences (two voices, three speed and pause settings), every written word was timed in 408 takes; in the remaining ones the untimed written “word” is an emoji, which is removed and not spoken. Word times were ordered without overlap in 414 takes, and the timings’ total matched the audio length in 414. Of 684 word cuts, 97.2% fell on a gap between spoken words; the remainder lie inside words that espeak-ng merged or expanded, where no gap exists. We did not evaluate timings against human-annotated alignments: the durations are those the audio was generated from, and in our checks speech recognition timestamps [11] drifted by up to a few hundred milliseconds within a sentence, which made them unsuitable as a reference.

Inference settings. Separately, we scaled each voice’s noise scale and noise width to 0.7 of the model’s own values to reduce audible trembling. With UTMOS [14] and Whisper-based clarity on three sentences per voice, the mean benchmark score of 13 English voices rose from 93.2 to 94.8 and predicted naturalness from 4.16 to 4.26 (of 5). The difference is smaller than the per-voice reproducibility of the score (3.5 points); one listener (the author) preferred the calmer setting.

Language	Category	n	raw	script	ours
English	address	6	51.4	43.2	46.0
English	cardinal	12	15.4	29.3	15.2
English	date	23	59.0	29.0	20.5
English	decimal	6	4.4	36.7	4.4
English	electronic	19	10.2	10.9	16.8
English	fraction	6	57.9	62.4	25.5
English	math	2	0.0	0.0	0.0
English	measure	11	41.6	41.4	34.6
English	money	38	77.8	26.5	13.3
English	ordinal	10	16.7	23.1	16.7
English	range	12	37.4	23.8	19.1
English	roman	1	60.0	60.0	0.0
English	serial	13	9.4	11.6	10.1
English	telephone	7	79.9	83.2	12.0
English	time	9	103.0	80.0	19.0
English	whitelist	2	4.2	0.0	20.0
English	word	23	3.6	5.4	1.5
Spanish	cardinal	37	51.2	15.5	46.8
Spanish	date	10	39.3	36.9	39.3
Spanish	decimal	10	25.2	28.6	16.6
Spanish	electronic	17	27.3	27.3	47.3
Spanish	fraction	40	66.6	64.3	47.6
Spanish	measure	19	49.2	55.9	39.6
Spanish	money	33	71.2	66.5	53.7
Spanish	ordinal	65	98.4	96.8	42.7
Spanish	telephone	6	70.2	69.7	28.6
Spanish	time	15	120.8	67.1	77.9
Spanish	whitelist	4	81.8	81.8	81.8
Spanish	word	21	0.0	0.0	0.0

Table 3: Mean PER (%) per NeMo category on the held-out half; the lowest value per row is in bold.

6 Limitations and threats to validity

The own suite was written by the author of the rules. NeMo’s cases are independent but follow one reading style, and *espeak-ng* reads each reference, so references it misreads (for example *one AM* read as the word *am*) add error to every system. Each case has a single reference. Full rules exist for English and Spanish only; other languages receive layout and pause rules. Heteronyms (*read* as present or past) and ambiguous forms (“3m”) remain. Large Spanish numbers are limited by *espeak-ng*. The American word list is curated and its benefit was checked at the phoneme level, not by listeners. No listening test has yet been run; one is in preparation. Synthesis is stochastic, and the timing and inference-setting results come from single takes.

7 Reproducibility and availability

The text preparation layer (package **ttsprep**, no dependencies) with its unit tests, an optional adapter that produces word timings with Piper, the evaluation scripts, our test suite with references, the NeMo cases with their source, licence and checksums, and all result files are released under the

Rule group switched off	English	Spanish
money	+12.6	+2.9
number grouping	0.0	+14.0
times	+3.8	+0.6
fractions	+1.0	+3.0
dates	+3.9	0.0
units	+0.5	+1.9
phone numbers	+2.4	-0.2
decades	+1.1	0.0
Roman numerals	+0.3	0.0
abbreviations	+0.2	0.0
ranges	+0.1	0.0
slashes	-0.3	0.0
web addresses	-0.5	-1.2

Table 4: Ablation on the held-out half: increase in mean PER (percentage points) when one rule group is switched off.

Apache License 2.0 in the study package at <https://loudness.ai/research/piper-tts-text-normalization/>. Running the two evaluation scripts reproduces the result files exactly. Piper and espeak-ng (GPL-3.0) are not included; the adapter uses an installed Piper at run time. Versions of all software are listed above. Voice models and NeMo’s code are not redistributed; they are available under their own licences.

8 Competing interests

The author develops Loudness.ai, whose text-to-speech tool uses the method described here. No external funding was received.

Version history. 1.0 (15 October 2026): first release. 1.0.1 (15 October 2026): corrects one misprinted character (the ordinal “10.^o” in Section 4); nothing else changed.

References

- [1] M. Hansen and the Open Home Foundation. Piper: a fast, local neural text to speech system. <https://github.com/OHF-Voice/piper1-gpl>.
- [2] eSpeak NG contributors. eSpeak NG text-to-speech. <https://github.com/espeak-ng/espeak-ng>.
- [3] J. Kim, J. Kong and J. Son. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *Proc. ICML*, 2021.
- [4] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf and C. Richards. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333, 2001.
- [5] P. Ebden and R. Sproat. The Kestrel TTS text normalization system. *Natural Language Engineering*, 21(3):333–353, 2015.
- [6] R. Sproat and N. Jaitly. RNN approaches to text normalization: a challenge. arXiv:1611.00068, 2016.
- [7] H. Zhang, R. Sproat, A. H. Ng, F. Stahlberg, X. Peng, K. Gorman and B. Roark. Neural models of text normalization for speech applications. *Computational Linguistics*, 45(2):293–337, 2019.

- [8] Y. Zhang, E. Bakhturina, K. Gorman and B. Ginsburg. NeMo inverse text normalization: from development to production. In *Proc. Interspeech*, 2021.
- [9] NVIDIA. NeMo text processing. <https://github.com/NVIDIA/NeMo-text-processing>, commit ddadfb2a38d2bc6b8cc6232c4f915eb60f500688.
- [10] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner and M. Sonderegger. Montreal Forced Aligner: trainable text-speech alignment using Kaldi. In *Proc. Interspeech*, 2017.
- [11] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey and I. Sutskever. Robust speech recognition via large-scale weak supervision. In *Proc. ICML*, 2023.
- [12] Y. Koizumi, H. Zen, S. Karita, Y. Ding, K. Yatabe, N. Morioka, M. Bacchiani, Y. Zhang, W. Han and A. Bapna. LibriTTS-R: a restored multi-speaker text-to-speech corpus. In *Proc. Interspeech*, 2023.
- [13] J. C. Wells. *Accents of English*. Cambridge University Press, 1982.
- [14] T. Saeki, D. Xin, W. Nakata, T. Koriyama, S. Takamichi and H. Saruwatari. UTMOS: UTokyo-SaruLab system for VoiceMOS Challenge 2022. In *Proc. Interspeech*, 2022.