

Enseñar a leer a un motor neuronal ligero de síntesis de voz: preparación de texto que conserva los fragmentos originales, corrección de pronunciación según el conjunto de fonemas y subtítulos cronometrados por el modelo en Piper

Asad Shehzad

Loudness.ai

Informe técnico LA-RB-2026-002, versión 1.0.1, 15 de octubre de 2026

<https://loudness.ai/es/research/piper-tts-text-normalization/>

Resumen

Piper es un motor neuronal de síntesis de voz muy utilizado que funciona en tiempo real en procesadores comunes. Lee el texto a través del fonetizador `espeak-ng` y no normaliza el texto por sí mismo, por lo que formas escritas como precios, fechas, abreviaturas, unidades y números de teléfono suelen leerse de forma literal. Describimos una capa de preparación de texto para Piper que reescribe esas formas y conserva, para cada palabra pronunciada, el fragmento del texto escrito del que procede; además, divide el texto en frases, líneas, párrafos y cláusulas con pausas explícitas. Presentamos dos observaciones más: varias voces comunitarias de inglés (EE. UU.) de Piper se entrenaron con el conjunto de fonemas británico de `espeak` y, por tanto, pronuncian palabras como *schedule* y *privacy* a la británica, lo que corregimos con cadenas de fonemas limitadas a los símbolos con los que se entrenaron esas voces; y la salida de duraciones de Piper proporciona tiempos exactos por palabra para subtítulos una vez que las palabras pronunciadas se asignan a las palabras escritas mediante una pequeña programación dinámica. En un conjunto de 84 frases de prueba en inglés y español escritas a mano, la tasa de error de fonemas frente a lecturas de referencia bajó del 32,6 % al 0,9 %, los finales de frase falsos bajaron de 18 a 0 y ninguna de las 19 frases de control cambió. En la mitad reservada de los casos de prueba públicos de normalización de texto de NVIDIA NeMo, que no escribimos nosotros, la tasa de error de fonemas fue del 41,0 % para el texto sin procesar, del 28,5 % para un script típico de números a palabras y del 15,8 % para nuestra capa en inglés (200 casos), y del 64,7 %, 56,2 % y 43,2 % en español (277 casos). Publicamos el código y los datos de evaluación bajo la licencia Apache 2.0.

1. Introducción

Los motores neuronales ligeros de síntesis de voz (TTS) hacen práctica la síntesis en servidores y dispositivos comunes. Piper [1], basado en VITS [3], es una opción frecuente: sus voces son pequeñas, funciona más rápido que el tiempo real con pocos núcleos de procesador y existen más de cien voces en decenas de idiomas. Piper convierte el texto en fonemas con `espeak-ng` [2] y sintetiza la voz frase a frase. No normaliza el texto, y las reglas propias de `espeak-ng` están pensadas para palabras y números sencillos. En la práctica, quienes despliegan Piper escriben sus propios scripts de preprocesamiento, normalmente en torno a una biblioteca que convierte números en palabras.

La normalización de texto para TTS es un área de investigación consolidada, desde la taxonomía de palabras no estándar de Sproat et al. [4] y los sistemas de producción basados en transductores

de estados finitos ponderados [5, 8] hasta los enfoques neuronales [6, 7]. Nuestro objetivo es más concreto y práctico: describir lo que debe hacer una capa de preparación pequeña y sin dependencias para Piper en particular, medir cuánto ayuda frente al texto sin procesar y frente a un script comunitario típico, e informar de dos problemas específicos de Piper que no encontramos documentados.

Nuestras contribuciones son:

1. Una medición de cómo lee `espeak-ng`, tal como lo usa Piper, las formas escritas cotidianas en inglés y español.
2. Una capa de reglas que conserva los fragmentos originales: cada regla reclama un fragmento del texto original, de modo que las reglas no pueden reescribir la salida de otras y cada palabra pronunciada puede rastrearse hasta la palabra escrita de la que procede; con segmentación en clases de pausa y uniones sin chasquidos.
3. La observación de que algunas voces comunitarias de inglés (EE. UU.) de Piper se entrenaron con el conjunto de fonemas británico de `espeak`, y una corrección que aporta pronunciaciones estadounidenses usando solo símbolos de ese conjunto.
4. Tiempos por palabra a partir de la salida de duraciones de Piper, asignados a las palabras escritas mediante programación dinámica, para subtítulos y resaltado por palabra; y la observación de que cargar en memoria el modelo que expone las duraciones casi duplica el uso de memoria frente a cargarlo desde un archivo.
5. Una evaluación con nuestro propio conjunto, con un conjunto independiente reservado, una ablación por grupos de reglas y una comprobación de la coherencia de los tiempos, con el código y los datos publicados.

2. Antecedentes

Piper y `espeak-ng`. Las voces de Piper son modelos VITS entrenados con secuencias de fonemas producidas por una voz concreta de `espeak-ng`, que la configuración de cada modelo registra (por ejemplo `en-us` o `en`). En la inferencia, Piper fonetiza el texto con esa misma voz de `espeak-ng`, lo divide en frases en los límites de cláusula de `espeak-ng`, asigna identificadores a los símbolos fonéticos y sintetiza cada frase. Los parámetros de inferencia controlan la velocidad de habla (*length scale*) y la variación aleatoria del sonido y del ritmo (*noise scale* y *noise width*). Piper acepta fonemas en bruto entre dobles corchetes. La versión 1.8 puede exponer las duraciones por fonema que predice el modelo marcando el tensor de duraciones como salida adicional.

Normalización de texto. La normalización convierte las palabras no estándar (números, fechas, moneda, abreviaturas, direcciones) en las palabras que diría un lector [4]. Los sistemas de producción usan gramáticas específicas de cada idioma compiladas en transductores de estados finitos ponderados [5, 8]; los sistemas neuronales aprenden la correspondencia, pero pueden producir errores irrecuperables [6, 7]. NVIDIA NeMo text processing publica gramáticas y casos de prueba para muchos idiomas [9]; usamos sus casos de prueba de inglés y español como datos de evaluación independientes.

Tiempos. Los tiempos por palabra de la voz sintetizada suelen obtenerse después de la síntesis mediante alineación forzada [10] o reconocimiento de voz con marcas de tiempo [11]. Un modelo no autorregresivo como VITS ya predice cuánto dura cada fonema, por lo que los tiempos pueden leerse del propio modelo.

3. Método

3.1. Reglas que conservan los fragmentos

Cada línea de texto pasa por una lista ordenada de reglas. Una regla es una expresión regular aplicada a la línea *original* junto con una función que devuelve lo que debe leerse en su lugar (o la rechaza). La primera regla que reclama un fragmento gana; las reglas posteriores no pueden coincidir dentro de un fragmento reclamado. Después, la línea se reconstruye a partir del texto sin cambios y de los reemplazos, y cada carácter del resultado lleva el fragmento del original del que procede. De ello se siguen dos propiedades: las reglas no pueden corromperse entre sí (una dirección de correo reclamada por la regla de direcciones nunca la tocan las reglas de abreviaturas o de barras), y cada palabra del texto preparado corresponde a un fragmento del texto escrito, de modo que *dollars* en “5 dollars and 50 cents” apunta a “\$5.50”. Una regla que conserva su primera palabra (“Felipe VI” → “Felipe sexto”) reclama solo el resto, de modo que cada palabra escrita conserva sus propias palabras pronunciadas. Los dobles corchetes que escribe el usuario se neutralizan para que no puedan inyectarse fonemas en bruto.

3.2. Cobertura

El inglés y el español tienen conjuntos de reglas completos: tratamientos y abreviaturas de calles (manteniendo el final de frase cuando una abreviatura cierra una frase, como en “Apple Inc. We...”), dinero con formas singulares y plurales y monedas regionales (“\$” se lee como pesos en México), fechas numéricas, ISO y con puntos en el orden de cada región, con años por pares en el inglés estadounidense (“twenty twenty-six”), horas con segundos, décadas, unidades, intervalos numéricos, números de teléfono en grupos de dígitos (incluido el formato de nueve dígitos de España), fracciones, números romanos tras palabras clave y nombres de monarcas, iniciales, millares agrupados con espacios y barras entre palabras. Todos los idiomas reciben las reglas de formato: se eliminan los marcadores de lista, el marcado markdown y los emojis, las direcciones web se acortan quitando el esquema y “www”, y las líneas sin puntuación final reciben un punto para terminar con entonación descendente.

3.3. Segmentación y pausas

El texto preparado se divide en segmentos, cada uno seguido de una clase de pausa: frase, línea de lista, párrafo, puntos suspensivos o cláusula. Los límites de frase se toman solo en un punto, un signo de interrogación o de exclamación no seguido de minúscula, una vez expandidas las abreviaturas. Los tramos sin puntuación de más de 28 palabras se dividen antes de una conjunción cercana a la mitad si existe (siete idiomas tienen listas de conjunciones) o, si no, en un espacio entre palabras, y la primera parte termina en coma para que la voz mantenga un tono de continuación. Con p como la pausa elegida por el usuario, los silencios son p tras una frase, una línea o unos puntos suspensivos, $0,4p$ tras una cláusula y $\max(2p, p + 0,3s)$ tras un párrafo. Cada segmento se sintetiza por separado y se une con silencios exactos. Como las voces suelen terminar en una

muestra distinta de cero (entre 28 y 57 dB por debajo del pico en nuestras mediciones), se aplica un fundido de entrada de 3 ms y uno de salida de 12 ms en cada unión, y se añade una cola de 120 ms.

3.4. Corrección de pronunciación para voces entrenadas con fonemas británicos

Varias voces comunitarias de inglés (EE. UU.), entrenadas desde cero con locutores estadounidenses de audiolibros (datos derivados de LibriTTS-R [12]), registran la voz **en** de *espeak*, que es inglés británico. Para la mayoría de las palabras esto es inocuo: el modelo aprendió a realizar los símbolos británicos con los sonidos estadounidenses de sus locutores. Importa cuando la forma británica es una pronunciación *léxica* distinta y no solo un acento distinto: *espeak en* da $ʃ'edju:l$ para *schedule*, $təm'ɑ:təʊ$ para *tomato*, $pɹ'ivəsi$ para *privacy* y $z'ed$ para la letra Z, y las voces dicen exactamente eso, mientras que **en-us** da $sk'edʒu:l$, $təm'eɪrəʊ$, $pɹ'arvəsi$ y $z'i:$. Cambiar estas voces a fonemas **en-us** no es una solución, porque sus datos de entrenamiento nunca contuvieron símbolos como $\mathfrak{x}$ o $\mathfrak{r}$. En su lugar escribimos formas estadounidenses de 118 palabras frecuentes con el conjunto de símbolos británico (incluidas palabras en *-ary* y *-ory*, palabras con pérdida de la yod [13] como *new* y *student*, y la letra Z) y las pasamos como bloques de fonemas en bruto solo a las voces estadounidenses cuyo modelo registra **en**. Una prueba unitaria comprueba que ningún bloque usa un símbolo ajeno al conjunto. Dos detalles de Piper importan: la puntuación situada justo después de un bloque en bruto se descarta y la palabra siguiente se une al bloque, por lo que la puntuación debe ir dentro del bloque; y un bloque en bruto seguido de texto deja un espacio doble, que eliminamos antes de sintetizar.

3.5. Tiempos por palabra a partir de las duraciones del modelo

Para cada segmento obtenemos, a partir de la salida de duraciones de Piper, la muestra inicial y final de cada símbolo fonético, incluidos los símbolos de puntuación, cuyas duraciones contienen las pausas. Las palabras pronunciadas se alinean con esta secuencia del modo siguiente. Sean $c_1, \dots, c_T$ el número de símbolos de cada palabra preparada cuando se fonetiza por separado, reescalados para que $\sum c_k = N$, el número de símbolos que no son espacios en la secuencia, y sea G el conjunto de posiciones donde empiece una palabra pronunciada. Elegimos los puntos de corte $0 = b_0 < b_1 < \dots < b_T = N$ que minimizan

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

con $\lambda = 1,5$ y cada b_k buscado a ocho símbolos como máximo de su posición proporcional, mediante programación dinámica. Esto tolera que *espeak-ng* fusione palabras cortas (“for the” se convirtió en una sola palabra pronunciada en nuestras pruebas) y que expanda una palabra escrita en varias (“2026”). Cada palabra preparada recibe el tiempo de sus símbolos; las palabras escritas reciben la unión de los tiempos de las palabras preparadas que apuntan a ellas, y un grupo de palabras preparadas que cubre varias palabras escritas (“5 euros con 50 céntimos” para “5,50 €”) reparte su tiempo en proporción a la longitud de las palabras escritas. Los subtítulos se construyen a partir de los segmentos: un subtítulo por frase o línea, los más largos divididos en espacios entre palabras para caber en dos líneas de unos 42 caracteres, y un subtítulo que termina a mitad de una idea (coma, puntos suspensivos, dos puntos) se une al siguiente cuando ambos caben.

Cargar el modelo con tiempos. Exponer el tensor de duraciones exige editar el grafo del modelo. El cargador de Piper puede hacerlo en memoria con el paquete **onnx**. En nuestro despliegue

esto usó 811 MB para cuatro voces frente a 480 MB cuando las mismas voces se cargaban desde archivos; escribir una vez en disco cada modelo editado y cargarlo desde el archivo devolvió el uso de memoria a 486 MB. Las copias editadas de los modelos de las 33 voces usadas en nuestra comprobación de publicación ocuparon 2,1 GB de disco.

4. Configuración experimental

Software. piper-tts 1.8.0 con su espeak-ng 1.52.0.1 incluido, onnxruntime 1.19.2, NumPy 2.0.2 y Python 3.9.6 en un portátil Apple M4 con macOS 26.6.1 para todas las mediciones de texto; onnxruntime 1.31.0, NumPy 2.5.4 y Python 3.12.3 en un servidor virtual AMD EPYC de 12 vCPU para las comprobaciones de producción. El script de referencia usa num2words 0.5.14.

Métrica. Para cada caso, la salida del sistema y una lectura de referencia se fonetizan con la misma voz de espeak-ng (**en-us** para el inglés de EE. UU., **en** para el inglés británico y **es** para el español). Tras eliminar las marcas de acento, la puntuación y los espacios entre palabras, la *tasa de error de fonemas* (PER) es la distancia de Levenshtein entre las dos cadenas de símbolos dividida por la longitud de la referencia; un caso es *exacto* cuando la PER es cero. Los *finales de frase falsos* cuentan la diferencia en el número de finales de frase descendentes (partes que terminan en punto, interrogación o exclamación). Como las referencias también las lee espeak-ng, la métrica aísla la normalización de la lectura que hace espeak-ng de las palabras comunes.

Conjunto propio. 103 frases (71 en inglés y 32 en español): 84 casos de prueba en 13 categorías con lecturas de referencia escritas a mano, y 19 frases de control de texto sencillo cuya referencia es el propio texto, de modo que cualquier cambio en un control es un perjuicio. Incluimos deliberadamente dos fallos conocidos (un heterónimo y una unidad ambigua).

Conjunto independiente. Casos de prueba de texto escrito a texto hablado en inglés y español de NVIDIA NeMo text processing en el commit `ddadfb2` [9] (957 casos utilizables). Los casos se dividieron en dos mitades mediante un hash del texto escrito. Tras una primera ejecución, estudiamos fallos individuales solo en la mitad de desarrollo y corregimos carencias generales (números agrupados con espacios, ordinales españoles escritos “10.^o”, precios españoles con el símbolo detrás del número, fechas con puntos y con el año primero, meses en minúscula, horas con segundos, una regla de markdown que eliminaba asteriscos entre dígitos). Una ablación en la mitad reservada mostró después que la regla de agrupación de números aumentaba ligeramente el error en inglés; lo investigamos solo con casos de desarrollo, lo que reveló que unía un día y un año (“june 20 2014”) y que no se reconocían las abreviaturas de mes en minúscula o con mayúscula inicial, y corregimos ambas cosas. Presentamos los resultados de la mitad reservada, cuyos casos individuales no inspeccionamos; la primera ejecución había mostrado puntuaciones agregadas por categoría sobre todos los casos antes de la división, y la ablación citada fue una señal agregada de la mitad reservada. Después de la evaluación, al revisar las recomendaciones de uso, vimos que los números de teléfono españoles en el formato agrupado de España (“912 345 678”) y las cantidades con millares agrupados con espacios antes de una moneda o unidad (“1 500 kg”) se unían en un solo número; corregimos ambas cosas y ninguna cifra publicada cambió. Las referencias de NeMo siguen su propio estilo de lectura, por lo que una lectura correcta en otro estilo (“two thousand twelve” frente a “twenty twelve” de NeMo, o nuestra omisión deliberada de “https://www.”) cuenta como error para todos los sistemas.

Grupo	n	PER (%)		Exactos		Finales falsos
		bruto	nuestra	bruto	nuestra	
Todos los casos de prueba	84	32,6	0,9	10	80	18→0
Inglés	58	33,6	0,9	6	55	13→0
Español	26	30,4	0,8	4	25	5→0
Controles (texto sin cambios)	19	0,0	0,0	19	19	0→0
nombre de marca	1	20,8	0,0	0	1	0→0
fechas	6	41,8	0,0	0	6	1→0
décadas	3	11,5	0,0	1	3	0→0
iniciales	2	0,0	0,0	2	2	3→0
límites conocidos	2	23,4	23,4	0	0	0→0
formato	7	72,6	0,0	2	7	3→0
dinero	9	50,5	0,0	0	9	0→0
teléfonos	3	85,2	0,0	0	3	0→0
intervalos, fracciones y barras	9	25,5	2,4	0	8	0→0
números romanos	5	26,7	0,0	0	5	0→0
horas	6	27,7	0,0	0	6	0→0
tratamientos	21	15,3	0,3	5	20	11→0
unidades	10	31,8	0,0	0	10	0→0

Tabla 1: Conjunto propio. PER frente a lecturas de referencia escritas a mano, número de casos exactamente correctos y finales de frase falsos (bruto→nuestra), sin preparación del texto (bruto) y con ella (nuestra).

Sistemas. *Bruto*: el texto tal como lo recibe Piper. *Script*: un preprocesador comunitario típico que escribimos para que fuera capaz y justo, combinando num2words con moneda, porcentajes, ordinales, años, decimales, fechas numéricas, horas, separadores de millares y abreviaturas comunes. *Nuestra*: la capa descrita arriba. El normalizador de NeMo no se evaluó con sus propios casos de prueba.

5. Resultados

Conjunto propio. La tabla 1 muestra que la capa eliminó casi todas las diferencias con las lecturas de referencia en las categorías a las que se dirige y no introdujo ningún cambio en las frases de control. Las cuatro diferencias restantes son los dos fallos deliberados, una fracción cuyo género en español la regla no puede conocer (*media taza*) y una variante ortográfica en una referencia (*Missus* por *Mrs*). Como escribimos tanto las reglas como este conjunto, estas cifras muestran la cobertura de los casos previstos, no la generalización.

Conjunto independiente. En la mitad reservada (tabla 2) nuestra capa tuvo el error más bajo en ambos idiomas. Por categoría (tabla 3), las mayores mejoras se dieron en dinero, números de teléfono, horas, fechas y fracciones. Nuestra capa fue peor que el texto bruto en las direcciones web, porque eliminamos el esquema y “www” a propósito, y el script fue mejor en los cardinales y las horas en español: espeak-ng lee números españoles grandes como 10^9 como *uno mil millones*, algo que el script evita al escribir los números él mismo, y NeMo lee *01:00* como *una* donde nosotros decimos *una en punto*.

Mitad	Idioma	n	PER (%)			Casos exactos	
			bruto	script	nuestra	bruto	nuestra
reservada	Inglés	200	41,0	28,5	15,8	65	99
reservada	Español	277	64,7	56,2	43,2	47	64
desarrollo	Inglés	221	41,9	29,4	19,5	68	106
desarrollo	Español	259	64,8	55,4	36,9	57	84

Tabla 2: Casos de prueba de NeMo. PER media del texto bruto, del script de números a palabras y de nuestra capa, y casos exactos. La mitad reservada no se inspeccionó; la mitad de desarrollo se muestra por transparencia.

Ablación. El tratamiento del dinero fue lo que más aportó en inglés y la agrupación de números lo que más aportó en español (tabla 4); un valor negativo significa que eliminar el grupo reduciría el error en este conjunto. Los grupos dirigidos a formas ausentes de los casos de NeMo (tratamientos, formato, pausas) no muestran efecto aquí y se evalúan con el conjunto propio.

Tiempos. En 414 tomas sintetizadas de las frases de prueba en inglés (dos voces, tres ajustes de velocidad y pausa), todas las palabras escritas recibieron un tiempo en 408 tomas; en las restantes, la “palabra” escrita sin tiempo es un emoji, que se elimina y no se pronuncia. Los tiempos de las palabras quedaron ordenados y sin solapamiento en 414 tomas, y su total coincidió con la duración del audio en 414. De 684 cortes entre palabras, el 97,2 % cayó en un espacio entre palabras pronunciadas; el resto está dentro de palabras que *espeak-ng* fusionó o expandió, donde no existe tal espacio. No evaluamos los tiempos frente a alineaciones anotadas por personas: las duraciones son las mismas con las que se generó el audio y, en nuestras comprobaciones, las marcas de tiempo del reconocimiento de voz [11] se desviaban hasta unos cientos de milisegundos dentro de una frase, lo que las hacía inadecuadas como referencia.

Ajustes de inferencia. Por separado, escalamos el *noise scale* y el *noise width* de cada voz a 0,7 veces los valores propios del modelo para reducir el temblor audible. Con UTMOS [14] y una medida de claridad basada en Whisper sobre tres frases por voz, la puntuación media de 13 voces en inglés subió de 93,2 a 94,8 y la naturalidad predicha de 4,16 a 4,26 (sobre 5). La diferencia es menor que la reproducibilidad de la puntuación por voz (3,5 puntos); un oyente (el autor) prefirió el ajuste más tranquilo.

6. Limitaciones y amenazas a la validez

El conjunto propio lo escribió el autor de las reglas. Los casos de NeMo son independientes, pero siguen un solo estilo de lectura, y *espeak-ng* lee cada referencia, por lo que las referencias que lee mal (por ejemplo *one AM* leído como la palabra *am*) añaden error a todos los sistemas. Cada caso tiene una sola referencia. Solo el inglés y el español tienen reglas completas; los demás idiomas reciben las reglas de formato y de pausas. Persisten los heterónimos (*read* en presente o en pasado) y las formas ambiguas (“3m”). Los números españoles grandes están limitados por *espeak-ng*. La lista de palabras estadounidenses es una selección y su beneficio se comprobó a nivel de fonemas, no con oyentes. Todavía no se ha realizado ninguna prueba de escucha; hay una en preparación. La síntesis es estocástica, y los resultados de tiempos y de ajustes de inferencia proceden de tomas únicas.

Idioma	Categoría	n	bruto	script	nuestra
Inglés	dirección	6	51,4	43,2	46,0
Inglés	cardinal	12	15,4	29,3	15,2
Inglés	fecha	23	59,0	29,0	20,5
Inglés	decimal	6	4,4	36,7	4,4
Inglés	electrónico	19	10,2	10,9	16,8
Inglés	fracción	6	57,9	62,4	25,5
Inglés	matemáticas	2	0,0	0,0	0,0
Inglés	medida	11	41,6	41,4	34,6
Inglés	dinero	38	77,8	26,5	13,3
Inglés	ordinal	10	16,7	23,1	16,7
Inglés	intervalo	12	37,4	23,8	19,1
Inglés	romano	1	60,0	60,0	0,0
Inglés	serie	13	9,4	11,6	10,1
Inglés	teléfono	7	79,9	83,2	12,0
Inglés	hora	9	103,0	80,0	19,0
Inglés	lista de términos	2	4,2	0,0	20,0
Inglés	palabra	23	3,6	5,4	1,5
Español	cardinal	37	51,2	15,5	46,8
Español	fecha	10	39,3	36,9	39,3
Español	decimal	10	25,2	28,6	16,6
Español	electrónico	17	27,3	27,3	47,3
Español	fracción	40	66,6	64,3	47,6
Español	medida	19	49,2	55,9	39,6
Español	dinero	33	71,2	66,5	53,7
Español	ordinal	65	98,4	96,8	42,7
Español	teléfono	6	70,2	69,7	28,6
Español	hora	15	120,8	67,1	77,9
Español	lista de términos	4	81,8	81,8	81,8
Español	palabra	21	0,0	0,0	0,0

Tabla 3: PER media (%) por categoría de NeMo en la mitad reservada; el valor más bajo de cada fila está en negrita.

7. Reproducibilidad y disponibilidad

La capa de preparación de texto (paquete `ttsprep`, sin dependencias) con sus pruebas unitarias, un adaptador opcional que produce tiempos por palabra con Piper, los scripts de evaluación, nuestro conjunto de pruebas con referencias, los casos de NeMo con su origen, licencia y sumas de verificación, y todos los archivos de resultados se publican bajo la licencia Apache 2.0 en el paquete del estudio en <https://loudness.ai/es/research/piper-tts-text-normalization/>. Ejecutar los dos scripts de evaluación reproduce exactamente los archivos de resultados. Piper y `espeak-ng` (GPL-3.0) no se incluyen; el adaptador usa un Piper instalado en tiempo de ejecución. Las versiones de todo el software se indican arriba. Los modelos de voz y el código de NeMo no se redistribuyen; están disponibles bajo sus propias licencias.

8. Conflictos de intereses

El autor desarrolla Loudness.ai, cuya herramienta de texto a voz usa el método descrito aquí. No se recibió financiación externa.

Grupo de reglas desactivado	Inglés	Español
dinero	+12,6	+2,9
agrupación de números	0,0	+14,0
horas	+3,8	+0,6
fracciones	+1,0	+3,0
fechas	+3,9	0,0
unidades	+0,5	+1,9
teléfonos	+2,4	-0,2
décadas	+1,1	0,0
números romanos	+0,3	0,0
abreviaturas	+0,2	0,0
intervalos	+0,1	0,0
barras	-0,3	0,0
direcciones web	-0,5	-1,2

Tabla 4: Ablación en la mitad reservada: aumento de la PER media (puntos porcentuales) al desactivar un grupo de reglas.

Historial de versiones. 1.0 (15 de octubre de 2026): primera versión. 1.0.1 (15 de octubre de 2026): corrige un carácter mal impreso (el ordinal “10.^o” de la sección 4); nada más cambia.

Referencias

- [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. En *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. En *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. En *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. En *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. En *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. En *Proc. Interspeech*, 2022.