PTEF: Um Framework Probabilístico para Estimar o Tempo de Pronúncia de Sequências Numéricas no Português Brasileiro

Adriano Carvalho dos Santos
Data Lead, EmDia — São Paulo, Brasil
Educação: MBA em Engenharia de Dados (2025–2026),
Universidade Presbiteriana Mackenzie;
Graduação em Banco de Dados (2021–2023),
Faculdade de Informática e Administração Paulista - FIAP
LinkedIn: linkedin.com/in/drico2236
Email: adriano.santos@magnasoluto.com.br

Setembro de 2025

Resumo

Apresentamos o *PTEF*, um framework probabilístico e computacional para estimar o tempo total de pronúncia da sequência $1, \ldots, N$ em português brasileiro. Diferentemente de somas com taxa fixa, o PTEF (i) modela a microduração por sílaba de forma *condicional ao contexto*, (ii) insere *pausas prosódicas* explícitas e (iii) realiza *contagem combinatória por blocos* em $O(\log N)$. O método retorna média e intervalo de confiança e está disponibilizado como software livre (Python e R), com testes e integração contínua. Aplicações incluem *length control* em TTS, verificação de alinhamento em ASR e curadoria de dados de fala.

1 Introdução

Estimar o tempo de contagem $1, \ldots, N$ em voz alta é útil para ensino, ergonomia vocal e validação de sistemas de fala. Abordagens com taxa de fala constante ignoram variações contextuais, pausas e fadiga; além disso, somar $1 \to N$ é custoso para N grande. Propomos o **PTEF**, que corrige essas fragilidades e se materializa em uma biblioteca de código aberto.

2 Formulação do PTEF

2.1 Decomposição de duração

Seja $p(n)$ a lista de tokens que pronunciam n:

$$T(N) = \sum_{n=1}^{N} \sum_{t \in p(n)} d\big(t \,|\, x_{n,t}\big) \; + \sum_{b \in B(N)} \pi(b). \tag{1}$$

onde $d(t \,|\, x)$ é a duração do token dado contexto local x, e $\pi(b)$ a duração de pausas em fronteiras prosódicas $B(N)$.

2.2 Microduração por sílaba (taxa variável)

Assumimos duração por sílaba lognormal e dependente do contexto:

$$\delta \sim \text{LogNormal}\big(\mu(x), \sigma^2\big), \quad \mu(x) = \beta^\top x, \tag{2}$$

$$\mathbb{E}[d(t) \mid x] = s(t) \cdot \exp\left(\mu(x) + \frac{\sigma^2}{2}\right). \tag{3}$$

O vetor x inclui posição no grupo, complexidade local, presença de palavras longas, efeito do falante e fadiga.

2.3 Pausas prosódicas

Pausas por classes (fraca/forte) com distribuições lognormais:

$$\pi(b) \sim \text{LogNormal}\big(\mu_b, \sigma_b^2\big), \quad b \in \{\text{fraca}, \text{forte}\}. \tag{4}$$

Na prática: respiração a cada B sílabas e pausa forte ao cruzar escalas (“mil”, “milhão/milhões”).

2.4 Forma fechada (esperança e variância)

Com vocabulário V e contagens $C_w(N)$ de cada token w na sequência $1, \ldots, N$:

$$\mathbb{E}[T(N)] = \sum_{w \in V} C_w(N)\, s(w)\, \mathbb{E}[\delta \mid x_w] + \sum_{c \in \{\text{fraca,forte}\}} C_c(N)\, \mathbb{E}[\pi \mid c], \tag{5}$$

$$\text{Var}[T(N)] \approx \sum_{w \in V} C_w(N)\, s(w)^2\, \text{Var}[\delta \mid x_w] + \sum_{c} C_c(N)\, \text{Var}[\pi \mid c]. \tag{6}$$

3 Gramática PB e política do "e"

Núcleo 0–999: unidades, dezenas especiais (10–19), dezenas (20–90), centenas ("cem" para 100; "cento" para 101–199); conector "e" entre centenas e resto e entre dezenas e unidades (quando o resto $\neq 0$). Para triades e escalas ("mil", "milhão/milhões"), adotamos a política **R1**: usar "e" entre escala e resto *se* o resto < 100; caso contrário, concatenar sem "e".

4 Contagem combinatória em $O(\log N)$

Evita-se enumerar $1 \to N$:

1. Pré-contar tokens e fronteiras para 1–999 (bloco base).
2. Decompor N em triades τ_j (base 1000) e replicar contribuições de blocos completos + pedaço parcial.
3. Somar escalas e conectivos "e" entre triades (segundo R1).
4. Estimar pausas: estruturais (ao cruzar triades/escalas) e respiração $C_{\text{resp}} \approx \left\lceil \dfrac{\sum_w C_w\, s(w)}{B} \right\rceil$.

5 Léxico de sílabas

O vocabulário é finito; usamos léxico PB com $s(w)$ obtido por IPA/regras (exemplo na Tabela 1).

Tabela 1: Amostra do léxico de tokens numéricos em PB com contagem de sílabas.

Token	$s(w)$	**Token**	$s(w)$
um	1	três	1
quatro	2	cinco	2
seis	1	sete	2
oito	2	nove	2
dez	1	onze	2
doze	2	treze	2
quatorze	3	quinze	2
dezesseis	4	dezessete	4
dezoito	3	dezenove	4

6 Disponibilidade e reprodutibilidade

O PTEF está publicado em **Python e R** no repositório público *MagnaSoluto/PTEF* [1], com:

- README com instalação, exemplos (API/CLI) e estrutura do projeto;
- Pacote Python (API e CLI via `ptef estimate`);
- Pacote R com funções equivalentes e vinheta;
- Dados de léxico, documentação técnica, integração contínua (CI) em Python e R;
- Testes automáticos cobrindo gramática, léxico, combinatória e integração.

7 Trabalhos futuros

Como evolução imediata planejamos:

1. Publicar o pacote Python no **PyPI** e o pacote R no **r-universe**/CRAN para instalação simplificada;
2. Executar validações experimentais com síntese de fala (TTS) e fala humana, reportando erros médios e ICs;
3. Expandir a gramática para escalas além de bilhões e permitir alternância pública da política do "e" (R1/R2).

8 Conclusão

O PTEF fornece uma estimativa eficiente ($O(\log N)$), com incerteza e aplicabilidade direta em sistemas de fala. A implementação aberta em Python e R, testes e CI aumentam a confiabilidade e a reprodutibilidade do método.

Agradecimentos

Esta formulação foi concebida pelo autor e desenvolvida/revisada com apoio de sistemas de inteligência artificial. Em particular, utilizaram-se o **Grok 4** (xAI) e o **GPT-5** (OpenAI) para suporte conceitual e revisão, e a plataforma **Cursor.ai** para auxiliar na geração e organização do código-fonte em Python e R.

Referências

[1] MagnaSoluto. *PTEF: Pronunciation-Time Estimation Framework*. Repositório GitHub. Disponível em: https://github.com/MagnaSoluto/PTEF. Acesso em: Set. 2025.

[2] M. McAuliffe et al. "Montreal Forced Aligner." *Interspeech*, 2017.

[3] Y. Wang et al. “Tacotron: Towards End-to-End Speech Synthesis.” *Interspeech*, 2017.

[4] D. Povey et al. “The Kaldi Speech Recognition Toolkit.” *ASRU*, 2011.