



Universidade Federal de Pernambuco
Centro de Informática
Departamento de Informática

Graduação em Ciência da Computação

Escansão automática de versos poéticos usando LLMs

André Arraes de Alencar Valença

Trabalho de Graduação

Recife
20 de março de 2024

Universidade Federal de Pernambuco
Centro de Informática
Departamento de Informática

André Arraes de Alencar Valença

Escansão automática de versos poéticos usando LLMs

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Departamento de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Filipe Carlos de Albuquerque Calegario*

Recife
20 de março de 2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

VALENCA, André Arraes de Alencar.

Escansão automática de versos poéticos usando LLMs / André Arraes de Alencar VALENCA. - Recife, 2024.

53 p. : il., tab.

Orientador(a): Filipe Carlos de Albuquerque Calegario

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Ciências da Computação - Bacharelado, 2024.

Inclui referências.

1. Large Language Models. 2. OpenAI. 3. GPT. 4. Escansão de Versos. 5. Poesia Popular. I. Calegario, Filipe Carlos de Albuquerque. (Orientação). II. Título.

000 CDD (22.ed.)

ANDRÉ ARRAES DE ALENCAR VALENÇA

ESCANSÃO AUTOMÁTICA DE VERSOS POÉTICOS USANDO LLMS

Trabalho de Conclusão de Curso apresentado à
Universidade Federal de Pernambuco - UFPE, como
requisito parcial para obtenção do título de Bacharel
em Ciência da Computação.

Data de aprovação: 20 de março de 2024

BANCA EXAMINADORA:

Prof. Filipe Carlos de Albuquerque Calegario

Prof. Giordano Ribeiro Eulálio Cabral

RECIFE,
2024

*Dedico este projeto aos cantadores e poetas do Pajeú, do
Cariri, de São Bento do Una e de todo interior nordestino,
por manter a letra viva na voz, e a voz viva através das
gerações. O gênio popular jamais será suplantado por
artifícios sintéticos (mas a gente tenta).*

Agradecimentos

Agradeço a meu orientador Filipe Calegario, pela crença, pelo estímulo e pela parceria.

A Bernardo, Daniel, Décio, Laura, Lígia, Nena e Oto, pelo apoio em mais uma presepada acadêmica.

Aos amigos Ceci, Leo, Guilherme, Pedro, Rafa Cruz e Rapha Prestes, pela atenção que me dedicam quando falo besteira.

A Rafael Nonato, por me fornecer uma literatura especializada sobre escansão poética.

Aos outros entes queridos, da academia e de fora, por serem razão para sair para o mundo.

Aos artistas e pesquisadores que nos precedem, sobre os ombros dos quais nos apoiamos.

*Sou Cantador mais falado,
Que pode existir no mundo;
Conheço qualquer matéria,
Sou lente muito jucundo;
Para falar a verdade,
Sou primeiro sem segundo!*

—RAUL FERREIRA (poeta popular de Vitória de Santo Antão)

Resumo

O presente trabalho tem o intuito de contribuir para a área de geração automática de textos em língua portuguesa, ao tentar construir um "escansor" de versos poéticos valendo-se do poder e capacidade de *large language models* da OpenAI, especificamente, o GPT-3.5 e o GPT-4. Os experimentos foram conduzidos utilizando técnicas como *zero shot*, *few shot prompting*, *chain of thought* e *fine-tuning*. O projeto como um todo integra-se a um plano mais amplo de desenvolver uma plataforma, chamada Pajeú, que fornecerá ferramentas de geração textual, treino literário e jogos de palavras, focados na poesia popular do Nordeste brasileiro.

Palavras-chave: Large Language Models, OpenAI, GPT, Escansão de Versos, Poesia Popular

Abstract

This work aims to contribute to the field of automatic text generation in Portuguese by attempting to construct a syllable separation tool for poetic verses using the power and capabilities of OpenAI's large language models, specifically GPT-3.5 and GPT-4. Experiments were conducted using techniques such as zero-shot, few-shot prompting, chain of thought, and fine-tuning. The project is part of a broader plan to develop a platform called Pajeú, which provides tools for text generation, literary training, and word games focused on popular poetry from the Northeast of Brazil.

Keywords: Large Language Models, OpenAI, GPT, Poetry Meter, Popular Poetry

Sumário

1	Introdução	1
1.1	Importância do projeto	1
1.2	Organização do trabalho	4
1.3	Objetivos	4
1.3.1	Objetivo Geral	4
1.3.2	Objetivos Específicos	4
2	Estado da Arte	5
2.1	LLMs	5
2.2	LLMs e textos poéticos	5
3	Escansão de versos poéticos	8
3.0.1	Abordagens para escansão automática	8
3.0.2	Metaplasmos	10
3.0.2.1	Sinalefa, sinérese	10
3.0.2.2	Dialefa, diérese	11
3.0.2.3	Aférese, síncope e apócope	11
3.0.2.4	Epêntese, paragore	11
3.0.2.5	Sístole, diástole	12
3.0.2.6	Eclipse	12
3.0.3	Ritmo	12
4	Metodologia e Resultados	14
4.0.1	Zero-shot prompting	14
4.0.2	Few-shot prompting	18
4.0.3	Chain of thought	20
4.0.4	Fine-tuning	23
4.0.4.1	Por quê fine-tuning?	23
4.0.4.2	Conjunto de dados	24
4.0.4.3	Treinamento	26
4.0.4.4	Performance	28
4.1	Custo	34
5	Conclusão	37
5.1	Observações finais	37
5.2	Trabalhos Futuros	39

CAPÍTULO 1

Introdução

1.1 Importância do projeto

Com avanços na área de Inteligência Artificial, houve sucessivos saltos qualitativos no campo do processamento de linguagem natural. Novas ferramentas, como os *large language models* (LLM) surgiram para auxiliar a criação textual, muito embora com o favorecimento da língua inglesa, dado, dentre outros fatores, ao seu extenso *corpus* disponível *online*.

A tarefa se torna ainda mais desafiadora quando se trata de textos estruturados, tais quais certos ramos da poesia. Enquanto existem os versos livres, popularizados por autores como Walt Whitman a partir de meados do século 19 (e pela nata modernista brasileira)(CUNHA; CINTRA, 2017, p.710), o verso metrificado permanece de forte expressão enquanto matéria prima de canções e poemas. No interior do Nordeste, por exemplo – entre poetas e cantadores – é a norma.

"[N]o mundo da cantoria, existem três pilares que precisam ser dominados pelo cantador: a rima, a métrica, a oração"(CALDAS, 2021), afirma reportagem do G1 sobre o "poeta do absurdo" Zé Limeira. A rima é o casamento fonético a partir da última sílaba tônica de uma palavra (ou verso); já a métrica se refere à divisão de sílabas poéticas de um verso, e seu padrão rítmico; enquanto que a oração é o assunto do poema, que em geral versa sobre uma um pensamento ou uma história. Ao produzir versos *nonsense* ("ano passado eu morri, mas esse ano eu não morro"), Zé Limeira se consagrou nas duas primeiras, porém não na última.

Analogicamente, é possível inverter os polos quando o tema é LLMs. Quando propriamente manejados (e quando não alucinam), modelos de linguagem avançados devolvem *outputs* com "orações" contextualizadas e compreensíveis. No entanto, por se tratar de um sistema preditor, em que o que está em jogo é a probabilidade do próximo *token*, há uma inerente dificuldade dos *large language models* em realizar tarefas de forma sistemática. O "átomo" dos LLMs é o *token*, uma divisão arbitrária feita a partir do processamento do *corpus* de treinamento, que identifica os pedaços mais comuns e os indexa. I.e., não é necessariamente a palavra, a sílaba, a letra, ou o fonema. O sucesso em dividir uma sílaba ou formar uma rima passa a ser regido por uma roleta, cada dia mais precisa, embora ainda vacilante.

Dadas estas questões, o presente trabalho tem o intuito de colaborar com a geração automática de textos em língua portuguesa. Pretende-se que essa seja apenas uma peça para auxiliar a construção de versos metrificados. Para isso, serão aplicadas diferentes abordagens de geração utilizando modelos de LLM da OpenAI. Mais especificamente, o objetivo é ensinar os modelos GPT-4 e/ou GPT-3.5 a fazerem possíveis escansões de versos poéticos (ato de metrificar e reconhecer o padrão rítmico de um verso) de maneira confiável.

Este projeto tem o escopo possível para funcionar como Trabalho de Conclusão de Curso

do Centro de Informática, mas faz parte de um plano de pesquisa mais expansivo, cujo intuito é de desenvolver uma plataforma de treino, geração textual e jogos literários com temática de poesia popular do Nordeste brasileiro, intitulada Pajeú.

Vale deixar claro que o presente projeto não coloca o termo "popular" em disputa, por se tratar de um trabalho mais especificamente sobre escansão. Entretanto, a saber, há uma larga discussão na teoria literária sobre a nomenclatura "(cultura) popular", e seu risco de desvalorizar essa produção artística que não faz parte do chamado "cânone". Argumenta-se também, por outro lado, que a manutenção do termo pode fortalecer seu impacto, por sublinhar a origem. Para entender as tensões semânticas em torno da expressão, basta levantar a querela em torno do Movimento Armorial, de Ariano Suassuna, que almejou unir o "erudito" ao "popular". Se olharmos pela ótica de uma tentativa de "elevar" o popular através de um decoro erudito, então há uma inferiorização de um em relação ao outro. Mas enxergando como tentativa de equiparação da cultura popular à erudita, há a exaltação de uma manifestação que deveria ser entendida como canônica, mas não é. Lourival Holanda, acerca deste tema, afirma: "Ariano vinha reivindicar os valores disseminados na cultura popular como valores reais. Queria, sobre eles, um olhar além de toda condescendência do olhar exótico". [p. 103]([HOLANDA, 2007](#)) Dentro disso tudo, ainda há a própria discussão sobre a relevância do "cânone" na literatura, que é um buraco ainda mais fundo. Então é melhor não nos estendermos mais. Todavia, esta pode ser uma pauta possível para outra etapa do projeto.

Voltando ao assunto. Pajeú é o rio que passa por cidades nordestinas com tradição em poesia popular, a mais conhecida das quais é São José do Egito-PE. De lá, saíram poetas como Cancão, Antônio Marinho, Lourival Batista (Loro do Pajeú), dentre muitos outros. A região no entorno do rio forma tantos representantes da arte do verso, que reza uma lenda de que aquele que beber as águas do Pajeú é agraciado com o dom da rima. No livro *Roteiro de velhos cantadores e poetas populares do Sertão* ([WILSON, 1986](#)), o pesquisador Luís Wilson argumenta que conceber poemas, principalmente no improviso, também é fruto de treino e memória extensivos.

Os glosadores de raça são ágeis de pensamento. São homens certamente bem dotados para sua função. Mas, também pacientemente exercitados na ginástica da cantoria. Se diante do público dão a impressão de improvisar – e de fato improvisam em muitas ocasiões, – isto acontece por causa de seu talento estar lavrando em materiais acumulados e elaborados mediante longas práticas e muita experiência. ([WILSON, 1986](#), p. 39)

Pensando por esse lado, a plataforma pretende ser um campo de treinamento para poetas exercitarem seu ofício. Os usuários poderão consultar rimas, gerar imagens inspiradoras para seus poemas, receber "motes" para encaixar em seus versos e "deixas" para continuá-los e, em seu modo mais avançado, deverá ser possível "pelejar" contra a máquina. Uma "peleja", ou "desafio", é um bate-e-volta entre dois poetas ou cantadores, no qual disputam um argumento, ou se deferem insultos, ou ambos. Abaixo, uma peleja entre os poetas paraibanos Antônio da Cruz e Cajarana, um cristão e um ateu, em torno da existência do divino. Cada um deles trabalha em cima de um mote de um verso, respectivamente "Deus é um Pai que nos cria!" e "Não creio que exista Deus". Quando há um mote, ele deve ser encaixado em toda estrofe do poema, em geral

no fim.

A.C.

Olhe para a amplidão
Que se vê no firmamento,
Ouça o murmúrio do vento,
Quando estremece o trovão!
O relâmpago e seu clarão,
Uma luz que alumia...
Dos morros à “Oceania”,
Desde o rio à correnteza;
Olhe para a natureza:
Deus é um Pai que nos cria!

C.

Eu quero cinco mil reis,
Não quero o Deus de vocês;
Ele, a vinte um do mês,
Expulsou-me a ponta-pés.
E eu não sou como Moisés,
Que enganou os cananeus;
Quase que mata os judeus,
Com falta de água e pão;
Vão fazer besta do cão.
Não creio que exista Deus!
(LINHARES; BATISTA, 1976, p. 106)

Como última peça do quebra-cabeça, deve-se convocar a funcionalidade de *low latency speech to speech* para preservar o caráter muitas vezes verbal dessas interações. A máquina seria capaz não apenas de captar a fala do usuário e convertê-la em texto, como também devolvê-la em áudio. Entende-se que, ao levantar essas funcionalidades, a plataforma funciona num terreno caudaloso entre o apoio a criação, a geração automática e a colaboração humano-máquina.

Ferramentas de suporte à criatividade aumentam e melhoram a criatividade humana, tais quais o Photoshop da Adobe ou ferramentas de auxílio computacional ao design. *Sistemas generativos* produzem artefatos criativos (semi-) autonomamente, tais quais computadores que fazem pinturas (veja Fig. 7.1) (McCorduck 1991; Colton and Wiggins 2012) or ou geram poesia (Colton et al. 2012). *Parceiros computacionais* colaboram com humanos em tarefas criativas tal qual um outro humano o faria. (DAVIS et al., 2015, p. 110, tradução nossa)

Isolado neste cenário, o escansor apresentado no presente trabalho funcionaria apenas como uma engrenagem para o controle de metrificação dos poemas. Se trata de uma peça de um mecanismo maior (Pajeú), que apresenta funcionalidades múltiplas, como soluções para escolhas de rimas, acesso a vocabulário típico da região e noções de como estruturar estrofes; assim como ser capaz de entender o contexto e responder de acordo.

1.2 Organização do trabalho

Esta seção apresenta a estrutura dos próximos capítulos, oferecendo um breve resumo de cada um deles.

- **Estado da arte:** neste capítulo, damos uma visão geral sobre o que são LLMs, e apresentamos algumas iniciativas de geração automática de textos "criativos" que usam a capacidade de modelos de linguagem em larga escala.
- **Escansão de versos poéticos:** são apresentadas regras para fazer a escansão de versos poéticos, os metaplasmos aplicados no processo e noções de ritmo. Também aprofundamos no procedimento de escansão automática do sistema Aoidos.
- **Metodologia e resultados:** nesta seção, são apresentados experimentos e resultados de escansão poética usando modelos da OpenAI, com abordagens de *prompt-engineering* tais quais *zero-shot prompting*, *few-shot prompting*) e *chain of thought*, assim como *fine-tuning*.
- **Conclusão:** concluímos o projeto com uma análise sobre os experimentos conduzidos, seus resultados e a relevância das métricas escolhidas. Também falamos dos possíveis rumos que o projeto pode tomar a partir daqui.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo deste trabalho é criar uma ferramenta com uso de *large language models* que faz escansões automáticas de versos poéticos, metrificando-os e exibindo seus padrões rítmicos.

1.3.2 Objetivos Específicos

- Escandir os versos via *zero shot prompting*.
- Escandir os versos via *few shot prompting*.
- Escandir os versos via *chain of thought*.
- Recolher dados de versos escandidos com propósito de fine-tuning.
- Escandir os versos via *fine-tuning*.

CAPÍTULO 2

Estado da Arte

2.1 LLMs

Modelos de linguagem de grande escala, ou *large language models* (LLMs), são um tipo específico de Inteligência Artificial. Assim como qualquer modelo de linguagem, ele transforma palavras, ou mesmo fragmentos de palavras (chamados *tokens*) em *embeddings* (vetores situados num espaço vetorial). Além de texto, LLMs podem trabalhar com outros tipos de dados sequenciais, como pixels, ou séries temporais. (LEE; TROTT, 2023)

Os LLMs são o resultado de um treinamento usando grandes bases de dados em uma arquitetura de rede chamada *Transformer*, que trabalha com dados sequenciais. Sua finalidade principal é aprender a distribuição de probabilidade do próximo *token* de uma sequência. Ao contrário de modelos de linguagem mais arcaicos, eles são capazes de "compreender" o contexto completo da sequência. (THOMPSON, 2023)

O GPT, modelo da OpenAI que atualmente já está na versão 4, é chamado de LLM *decoder-only*. Os "*decoders* são úteis para realizar tarefas em que um texto precisa ser completado (como completar uma sequência)" (THOMPSON, 2023, tradução nossa). Tanto *encoder-only* quanto os *encoder-only* são classes de LLM comumente chamadas de "generativa" (ou gerativa, ou geradora).

As IAs generativas entraram no léxico popular com o lançamento, em novembro de 2022, do ChatGPT, plataforma da OpenAI que permite um usuário conversar com uma LLM de propósitos gerais. O ChatGPT é uma instância do GPT treinada em conversas humanas reais e, em seguida, treinada utilizando aprendizado reforçado com *feedback* humano.

2.2 LLMs e textos poéticos

Embora modelos de linguagem em larga escala ainda estejam engatinhando no seu apelo popular, já é evidente a sua proficiência em gerar textos em linguagem natural muitas vezes indistinguíveis daqueles escritos por humanos. Determinados estudos, inclusive, já mostram que o ChatGPT é capaz de escrever redações dissertativas que seriam avaliadas mais positivamente por professores do que a média de alunos secundaristas. (HERBOLD et al., 2023)

Além da geração de dissertações compreensíveis, o potencial "criativo" de tais máquinas abre espaço para experimentações de cunho literário. O TextFX¹, ferramenta criada pelo Google Creative Lab em parceria com o rapper Lupe Fiasco, utiliza a API da PaLM 2 para brincar sintática e semanticamente com palavras e expressões. Os criadores fazem uso da aborda-

¹<https://textfx.withgoogle.com/>

gem de *few-shot prompting* para desenvolver acrônimos, aliterações, cadeias de pensamento, expressões com mesma similaridade sonora, dentre outros jogos literários. (WADE, 2023)

Para se ter uma ideia de como o TextFX funciona, há uma ferramenta chamada "Explode", que devolve espécies de trocadilhos. Ao inserir uma palavra, o Explode a quebra em outras menores que, juntas, soam como a primeira. Ao se inserir a palavra "arara", por exemplo, uma das respostas devolvidas seria "air are a (*a bird that breathes air*)". O fascinante do TextFX é que ele usa de uma abordagem das mais simples para extrair um resultado rico, e com propósito claro. Inclusive, consegue extrair a fonética das palavras a partir da ortografia (e olha que, no inglês, essas duas coisas são bem apartadas).

PoeLM² é um modelo de linguagem para geração automática de poemas em Espanhol e Basco com métrica e estrofes estruturadas. A heurística escolhida foi a de pegar um *corpus* não poético e, a partir de marcadores de pontuação, dividi-lo em frases. A essas frases são atribuídos códigos de controle que informam seu tamanho e o final da última palavra (fazendo as vezes da métrica e a rima). Ao fazer uma requisição, em tempo de inferência, é aplicado um descritor de estrutura que condiciona a resposta a seguir o padrão do treinamento. O melhor retorno é filtrado diante de variadas opções. (ORMAZABAL et al., 2022) No treinamento do nosso projeto, usamos uma abordagem parecida, no sentido que colocamos marcadores gráficos nos exemplos, assim como o PoeLM usa códigos de controle.

O CoPoet³ (CHAKRABARTY; PADMAKUMAR; HE, 2022) usa vários modelos T5 (Text-to-Text Transfer Transformer) para construir um sistema colaborativo de escrita poética. O âmago do sistema é um modelo de linguagem treinado num *corpus* com instruções para escrever poemas. A solução é interessante, porque promove uma colaboração entre o humano e a máquina em prol de um objetivo comum, que é escrever um poema. Tem um caráter didático, ao tentar "afloorar" a criatividade no usuário. O *corpus* de treinamento também se destaca, por se tratar de um par <instrução, verso>. A instrução pode ser figuras de linguagem como metáforas, onomatopeias; ou rimas, assuntos, e outros procedimentos relevantes ao se escrever uma poesia.

O Poet Vicuna é uma experimentação neste campo que se vale do LLaMA-13B para criar versos metrificados em inglês. Para isso, foi construída uma classe chamada "MetricGenerator", que "controla a estrutura métrica do *output* gerado ao restringir o a distribuição de probabilidade do modelo sobre os tokens". (HOOVER, 2023, tradução nossa) Essa abordagem usa, portanto, "filtros" e funções fora do contexto da LLM, e é mais possível em modelos *open source*. A plataforma⁴, no entanto, tem um ar muito "mecanizado", com muitos parâmetros a serem preenchidos, o que pode desestimular o usuário.

Mesmo com grandes avanços neste quinhão específico das IAs generativas, a geração automática de textos literários especificamente em língua portuguesa ainda tem um longo caminho a percorrer. Evidências recentes mostram resultados relevantes no uso de LLMs (treinados em línguas e *corpora* diversos) como base para refinar a geração de conteúdo num idioma ou domínio específico. O Sabiá (PIRES et al., 2023), modelo treinado em cima do GPT-J e LLaMA, é uma recente incursão na tentativa de "abrasileirar" grandes modelos, além de enfa-

²https://github.com/aitormazabalpoetry_generation

³<https://copoet-emnlp.github.io/>

⁴<https://replicate.com/replicate/poet-vicuna-13b>

tizar aspectos históricos, geográficos e culturais-específicos do Brasil que, grosso modo, ficam "enterrados" diante do volume de informação anglófona. O Sabiá também evidencia que pode ser mais barato trabalhar em cima de modelos grandes, já pré-treinados, mesmo que em outras línguas. Seus resultados demonstram que o custo para treiná-lo representa menos de 3% do orçamento necessário para pré-treinar os modelos em cima dos quais trabalha.

Voltando algumas casas, torna-se patente que se aventurar na mistura de literatura com computação não é uma prática tão nova. Abordagens mais estruturadas, até mesmo algorítmicas, já surgiram para resolver uma série de questões da área. Desde o primitivo e pitoresco *Gerador de lero lero*⁵, que gera textos aleatórios pseudo-intelectuais, até, por exemplo, uma ferramenta de escansão (ato de separar as sílabas poéticas de um verso) chamada Aoidos, que surge em 2016 como forma de sistematizar a divisão silábica e aplicar as tônicas do verso.

A plataforma Aoidos, aliás, é uma pedra fundamental para o presente projeto, pois através dela foi gerado o *corpus* de textos corretamente escandidos que alimentam a estratégia de executar fine-tuning do modelo GPT-3.5 com milhares de exemplos. É um passo atrás para dar dois passos à frente.

⁵<https://lerolero.bgnweb.com.br/>

CAPÍTULO 3

Escansão de versos poéticos

Para se ter uma ideia da tarefa que o LLM deve executar no presente projeto, é necessário primeiro entender no que se consiste a escansão de versos poéticos. "[E]scandir um verso significa dividi-lo em sílabas de forma a destacar seu metro e seu ritmo; ao verbo escandir corresponde o substantivo escansão."(MITTMANN, 2016, p. 1) O "padrão rítmico", ou "esquema rítmico", determina as tônicas do verso, e cria, através da alternância de sílabas tônicas e átonas, subidas e descidas no tom da pronúncia. Já a métrica é a divisão silábica poética em si, que difere da divisão gramatical em alguns aspectos. Um exemplo:

- nos verdes ramos -> nos / * ver- / des / * ra- # / mos

O verso "nos verdes ramos" é dividido em 5 sílabas, com tônicas em "ver" e "ra", representadas pelo sinal gráfico *. Ele é um tetrassílabo, ou seja, apresenta 4 sílabas poéticas, pois todas as sílabas que vêm após a última tônica (neste caso, apenas "mos") não contam para a metrificação. A última sílaba válida é representada aqui, além do *, pelo sinal gráfico #. Este verso é escandido de forma não-ambígua, pois não há mais de uma forma de separar as sílabas poéticas que não a apresentada acima. Mas isto não é verdade para todo verso.

Escandir significa dividir o verso em sílabas poéticas. Note que nem sempre as sílabas poéticas correspondem às sílabas gramaticais. O leitor-ouvinte pode juntar (ou separar) sílabas, quando houver encontro de vogais, de acordo com a melodia do verso. O ouvido de cada um vai indicar como proceder. (GOLDSTEIN, 1988, p. 16)

Ou seja, embora haja um caráter algorítmico na divisão silábica, também há algo de contextual e cultural-específico e, portanto, subjetivo. Um verso que talvez a maioria das pessoas escandisse com seis sílabas, dentro de uma estrofe de sete sílabas seria "forçado" a ser pronunciado com as sete, de acordo com regras (chamadas "metaplasmos") que podem ser aplicadas. Uma palavra que tenha determinada pronúncia no português brasileiro (ou português de determinada região do Brasil), talvez seja pronunciada naturalmente de outra maneira no português de Portugal, com a determinação de juntar ou separar a pronúncia de alguns hiatos. Todos esses metaplasmos são opcionais, e a escolha de como aplicá-los é prerrogativa do poeta e/ou do leitor.

3.0.1 Abordagens para escansão automática

Não obstante não haver uma forma determinística de se escandir, o presente projeto se propõe a usar de modelos de linguagem para receber um verso como *input (prompt)* e devolver

o verso escandido de maneira aceitável como *output (response)*. A metrficação do verso deve ser acompanhada de um indicativo de padrão rítmico. No formato assumido pelo projeto, as sílabas poéticas são separadas pelo sinal gráfico "/", as tônicas do verso são precedidas por "*", e a última sílaba forte é sucedida por "#".

Para atingir tal resultado, testou-se quatro abordagens diferentes. A primeira, vale-se exclusivamente de um prompt descontextualizado pedindo a escansão de versos poéticos. A segunda faz uso da abordagem de *few-shot prompting*. A terceira tenta criar, da forma mais resumida possível, uma *chain of thought*. A última faz uso do *fine-tuning* com uma base de dados coletada e gerada pelo Aoidos.

O Aoidos é um sistema desenvolvido por Adiel Mittmann como resultado de sua tese de doutorado em Ciência da Computação pela Universidade Federal de Santa Catarina (UFSC). Ele leva em conta todas as peculiaridades descritas acima sobre o processo de escansão, e mais algumas, numa abordagem algorítmica. A ferramenta foi criada com o intuito de escandir versos e estrofes poéticas automaticamente e, por exemplo, ao receber como *input* (antes das setas) a seguinte estrofe de dois versos, devolve o *output* (após as setas):

- o pato lamenta quando quer -> o / pa / to / la / men / ta / quan / do / quer #
- a patinha de quem tem saudades -> a / pa / ti / nha / de / quem / tem / sau / da # / des

O primeiro verso tem, sem ambiguidades, 9 sílabas poéticas, terminando na tônica “quer”. O segundo verso é escandido de forma a assumir um metaplasmo chamado “sinérese” na palavra “saudades”, em que o ditongo “sau” permanece como ditongo, mantendo a consistência das 9 sílabas. No entanto, ao colocar a estrofe,

- o pato lamenta quando quer ver -> o / pa / to / la / men / ta / quan / do / quer / ver #
- a patinha de quem tem saudades -> a / pa / ti / nha / de / quem / tem / sa / u / da # / des

agora o primeiro verso contém dez sílabas poéticas, de maneira não-ambígua, com o término em “ver”. Desta vez, o sistema força uma diérese na palavra “saudade”, escandindo-o como “sa / u / da / de”, para que a estrofe mantenha-se como decassílabo.

O Aoidos usa de uma conversão de grafema para fonema para determinar as sílabas, e tem diversos critérios de “naturalidade” da enunciação. Ao escandir um poema, ele cria uma elocução base para cada verso, determina por uma série de parâmetros qual seria a metrficação “dominante” da estrofe e passa por uma nova escansão, aplicando-se regras que “desnaturalizam” a escansão do verso em prol de se adequar à metrficação dominante e seu padrão rítmico. Como explica o autor,

As elocuições possuem um campo, *pts*, que tenta capturar a naturalidade da pronúncia por ela especificada. Na elocução básica, produzida a partir da junção das palavras individuais de um verso, o valor de *pts* é zero. Este não é um valor neutro de naturalidade, é um valor inicial. Esta pontuação, com efeito, é um valor relativo, não absoluto. Comparar a pontuação de duas elocuições produzidas para o mesmo verso faz sentido, mas comparar pontuações de elocuições oriundas de versos diferentes não. (MITTMANN, 2016, p. 15)

Em outras palavras, o sistema tem regras de naturalidade baseadas numa padronização da pronúncia do português que são admitidamente "subjettivas" (MITTMANN; MAIA, 2017, p. 162), embora haja uma tentativa de se universalizar. Elas vão perdendo esse ideal de naturalidade à medida que têm de se adequar a padrões rítmicos da frase. Aplicar determinados metaplasmos em determinadas situações contribuem para a naturalidade da pronúncia, enquanto outras vezes a estorvam. Abaixo, segue uma breve explicação desses recursos; uma versão simplificada de como estão descritos na tese de Mittmann, e que servem de critério para a escansão produzida pelo Aoidos.

3.0.2 Metaplasmos

Metaplasmos são fenômenos fonéticos que ocorrem quando versos são escandidos. Alguns desses fenômenos são forçosamente aplicados com o intuito de preservar a metrificação de um verso. Abaixo, são apresentados os metaplasmos considerados pelo Aoidos ao escandir um verso, com uma explicação sucinta do seu efeito.

3.0.2.1 Sinalefa, sinérese

O metaplasmo que ocorre com mais frequência ao se escandir um verso é a sinalefa. Em suma, trata-se da junção de sons vocálicos que seriam separados pelo fato de estarem em palavras distintas. Alguns exemplos:

- sobre os vivos -> so- / bre os / vi / vos

Percebe-se, ao ler em voz alta, que o fim da palavra “sobre” e o início de “os” juntam-se em uma só sílaba poética, “bre os”. Isto acontece porque basta um impulso vocálico para pronunciar esses dois trechos.

- Estarei morto?! E a esta pergunta -> es / ta / rei / mor / to?! e a es- / ta / per / gun / ta

O exemplo b) ocorre similar ao exemplo a), embora de maneira mais radical. Quatro vogais se unem para formar um único impulso vocálico.

A sinérese é o equivalente da sinalefa no contexto de uma palavra, e não de um bigrama. Um exemplo é quando se junta as vogais de um hiato num só som, transformando-os em ditongos. É o que ocorre com palavras como “navio”, um hiato com a divisão silábica “na-vi-o”, que pode ser escandida como “na / vio”.

Observação: chama-se “elisão”, quando a sinalefa é tão eficiente que “remove” foneticamente a vogal de uma das palavras conjugadas.

- ela é forte -> e- / lá é / for- / te

No exemplo c), a vogal “a” de “lá é” deixa de existir na pronúncia quando há a sinalefa, promovendo uma elisão.

3.0.2.2 Dialefa, diérese

É o oposto da sinalefa. Quando, por desígnio próprio, opta-se por separar as sílabas que constituem um hiato, ou um ditongo, em geral para alcançar uma métrica específica do verso. Via de regra, essa aplicação "desnaturaliza" a pronúncia, mas nem sempre.

- Deus e anjo -> Deu- / s, e / an- / jo

Neste caso, separa-se o “e” do “an” na divisão poética, que poderia apresentar sinalefa. No escopo de uma palavra, e não do verso como um todo, tem-se a diérese. Exemplos como “ vaidade ” (“vai-da-de”) podem ser divididos na poesia como “va / i / da / de”, apresentando dialefa.

3.0.2.3 Aférese, síncope e apócope

Aférese é a remoção de uma vogal no início da palavra. A palavra “estuda” é passível de se pronunciada “studa”, eliminando o “e”. Então, podemos dividir o verso assim:

- estuda -> Estu / da

Na síncope, remove-se uma vogal do meio da palavra. Palavras como “pétala” facilmente podem ser pronunciadas como “pétla”, removendo-se o primeiro “a”. A divisão ficaria assim:

- pétala -> pé / tala

Na apócope, remove-se o final de uma palavra. A palavra “mármore”, por exemplo, pode perder o “o” e ser lida como “mármre”.

- mármore -> már- / more

3.0.2.4 Epêntese, paragore

É o uso de vogais de “apoio” em determinadas palavras. Quando se trata de uma união consonantal como “gn”, “ct”, ou “bs”, dá-se o nome de epêntese. “Obscurecida” se lê “obiscurecida”.

- obscurecidas -> o- / bs- / cu- / re- / ci- / das

Quando se dá no fim da palavra, como com o “b” que finaliza a palavra “sob”, diz-se tratar de uma paragore. “Sob” se lê “sobe” (com o “o” fechado).

- sob -> so- / b

3.0.2.5 Sístole, diástole

Quando a tônica é modificada na palavra. Se a tônica vai mais para o começo da palavra, é uma sístole, mais para o fim, é uma diástole. O verso

- nem harmonia mais rara -> nem h- / ar- / * mo- / nia / mais / ra- / ra

faz a opção de pronunciar “harmonia” como “harmônia” para manter o padrão rítmico de um poema de Delminda Silveira.

Caetano Veloso canta, na canção *Tigresa*, “Como é bom poder tocar um instrumento!”, jogando a tônica de “instrumento” de “men” para “to”, lendo-se “instrumentu”. Isto se configura uma diástole. A separação silábica se mantém, mas assim ele cresce uma sílaba poética, de 11 para 12.

- ins / tru / * men # / to -> ins / * tru / men / * to #

3.0.2.6 Eclipse

A eclipse é o ocultamento de sons nasais. A palavra “virgem” pode assumir a forma “virge” não só como uma contração, mas como uma eclipse. Muitas vezes, o poeta deixa a eclipse explícita.

- Que virgem é essa -> Que / vir / gem é / e- / ssa (ou “Que / Vir- / ge’ é e- / ssa”)

3.0.3 Ritmo

Alterações (deliberadas ou não) na noção de ritmo também aparecem nos metaplasmos, como no caso mencionado das sístoles e diástoles, que alteram a posição da tônica sem se valer de artifícios para separar ou juntar sílabas. Uma forma básica de definir o esquema rítmico é dizer que se trata de uma recorrência de sílabas tônicas separadas por sílabas átonas. Segundo a *Nova gramática do português contemporâneo*, “[e]sta sucessão de sílabas fortes e fracas, com intervalos regulares, ou não muito espaçados (para que a reiteração possa ser esperada e sentida pelo nosso ouvido), é uma fonte do prazer a que chamamos ritmo.” (CUNHA; CINTRA, 2017, p.685)

O verso de Cruz e Souza

- * Vai, / Pe / re / * gri / no / do / ca / * mi / nho / * san # / to,

apresenta uma coincidência entre as tônicas de cada palavra com as tônicas do verso. Algumas palavras, no entanto, têm tônicas que não são fortes, como os monossílabos átonos (“te”, “pra”, “do”, entre outros). Via de regra, elas não são contadas, a não ser que haja uma intencionalidade do poeta, a partir da definição do esquema rítmico do poema como um todo, que a tonifique. O oposto também pode ocorrer, sílabas tônicas atonificarem em decorrência de alguns eventos. O mais comum deles é a atonificação por posição, em que duas sílabas tônicas se encontram, e uma delas ganha a batalha, em geral, a posterior.

O verso de Camões

- * Que / da o- / ci- / den- / tal / * prai- / a / Lu- / si- / * ta- # / na,

apresenta uma colisão entre as tônicas "tal" e "prai", na qual se sobressai "prai". Essa supressão é essencial à pronúncia oral, visto que a

contiguidade de sílabas tônicas prejudica o ritmo e, conseqüentemente, desagradada ao ouvido. Por isso, a sílaba anterior à última tônica é necessariamente átona. Tão forte é esta exigência rítmica que, mesmo sendo tônica no vocábulo isolado, ela se atonifica pela posição. (CUNHA; CINTRA, 2017, p. 685)

No sistema Aoidos, diversas checagens de sílabas tônicas a átonas são feitas antes mesmo da metrificação. As tônicas no verso são chamadas de "ictos", e, no português, há muita correspondência entre as tônicas ortográficas das palavras e os ictos. Assim como para a metrificação, o sistema tenta encontrar uma naturalidade (subjativa) para esse processo, seguindo ao máximo regras do português brasileiro, e aplicando devidas excessões. O sistema, então, gera algumas opções de ritmo. Há um reposicionamento dos ictos de acordo com o contexto da estrofe como um todo, com o intuito de guiar a leitura para o padrão provavelmente intencionado pelo poeta. Ao contrário de como ocorre com a determinação da métrica, esse processo não é feito a partir da fonética, e sim da ortografia. Como comenta o autor:

Estas regras foram estabelecidas pelo próprio autor da tese tratando os textos escritos segundo a ortografia vigente [...]. [E]m vez de buscar possíveis manuais que ensinem como saber onde está a vogal tônica de uma palavra, optou-se por observar as palavras do português e deduzir como o sistema ortográfico funciona. (MITTMANN, 2016, p. 90)

Outro tipo de adequação também é feito. Certos esquemas rítmicos são tradicionais na literatura de língua portuguesa, como é o caso do verso heroico (apresenta tônicas obrigatoriamente nas 6ª e 10ª sílaba, podendo ter até duas tônicas a mais de apoio); do martelo (3ª, 6ª e 10ª); do sáfico (4ª, 8ª e 10ª); dentre outros.

O sistema apresenta um catálogo de ritmos mais comuns a cada tamanho de verso. Por exemplo, um eneassilábico, de 9 sílabas, possui dois esquemas mais tradicionais: com as tônicas na 4ª e 9ª, ou com as tônicas na 6ª e 9ª. Ao primeiro é aplicado um fator de 0,6655 e, ao segundo, de 0,2513. Isto indica que este é menos significativo (em naturalidade) do que o anterior. Portanto, no contexto de uma estrofe completa, o sistema dará pontuações mais altas a opções de padrão rítmico canônicos. O sistema pode-se abdicar de usar o catálogo de ritmos e optar por uma abordagem mais conservadora que leva em consideração as tônicas.

Metodologia e Resultados

No experimento para escansão automática proposto pelo presente projeto, testamos quatro métodos diferentes em dois modelos de LLM: *zero-shot prompting*, *few-shot prompting*, *chain of thought* e *fine-tuning*. As primeiras três formas foram observadas em caráter ilustrativo, com um conjunto reduzido de dados e uma conferência manual. O intuito era ter uma *baseline* da capacidade do LLM de executar tal atividade. A primeira utilizou os modelos GPT-3.5-turbo-0125 e GPT-4-turbo-preview. A segunda e a terceira, apenas o GPT-4-turbo-preview. A última forma apresenta largo conjunto de dados e métricas objetivas para análise quantitativa e qualitativa, utilizando apenas o modelo GPT-3.5-turbo-0125.

4.0.1 Zero-shot prompting

Trocando em miúdos, *zero-shot prompting* é o puro ato de fazer uma requisição a uma LLM. É um *prompt* escrito com instruções para a máquina, da qual se espera uma resposta. Se trata da maneira mais comum de se interagir, por exemplo, com o ChatGPT, com o qual se tem conversas que levam a uma réplica do modelo.

Outro aspecto de se fazer uma requisição a uma LLM é a configuração de hiperparâmetros. Há hiperparâmetros que se podem ajustar num modelo os quais direcionam suas resposta para lugares mais diversos. A documentação do OpenAI revela alguns dos hiperparâmetros que podem ser alterados por usuários, como *temperature*, *maximum length*, *stop sequences*, *top P*, *frequency penalty* e *presence penalty*. Talvez o hiperparâmetro mais popular é o da temperatura, que interfere na previsibilidade da resposta do modelo, e o único que foi deliberadamente alterado do *default* nestes experimentos. Ela

controla a aleatoriedade do *output* do modelo em tempo de inferência. [...]. Uma temperatura mais baixa leva a um *output* mais determinístico da LLM, assim favorecendo as predições mais prováveis. [...] Uma temperatura mais alta devolve um *output* mais aleatório, assim favorecendo a predições mais "criativas". Essa criatividade é capturada pela predisposição do modelo de explorar respostas menos convencionais e menos improváveis. (RENZE; GUVEN, 2024, tradução nossa)

Como o intuito é realizar uma tarefa com um objetivo determinístico, para o experimento com *zero-shot prompting*, alteramos a temperatura do padrão no OpenAI Playgrounds de 0.7 para 0.2, tanto para o modelo GPT-3.5-turbo-0125 quanto o GPT-4-turbo-preview. Uma outra estratégia adotada foi seguir um roteiro na mensagem ao sistema que tenta informar à máquina quatro pontos relevantes:

- Papel - Qual personalidade, ou função, a máquina deve adotar;
- Contexto - O propósito da requisição do usuário;
- Instrução - O que a máquina deve de fato executar;
- Formato de saída - Que tipo de configuração a resposta da máquina deve adotar.

Numa primeira tentativa, foi suprimida a formatação, afim de entender, de forma mais pura, como a máquina faria a escansão dos versos. Segue abaixo o primeiro *prompt* para o modelo 4:

Você é um excelente assistente de poeta.

Eu preciso fazer a escansão de alguns versos (escansão é a
→ divisão silábica de versos poéticos e identificação das
→ tônicas).

Quando eu colocar um poema como input, quero que você o
→ devolva cada verso metrificado e com seu padrão rítmico
→ assinalado. Ou seja, o verso deve ficar dividido em
→ sílabas poéticas, com as tônicas dos versos marcadas.

A entrada do usuário deveria, portanto, se tratar de um poema a ser escandido. Escolheu-se a mais famosa sextilha de Cancão, poeta popular de São José do Egito. Como o intuito inicial foi o de observar o comportamento da máquina sem a preocupação de recolher métricas acerca de sua performance, pequenos *inputs* se demonstram suficientes para uma análise mais superficial. Eis o poema:

Centenas de pirilampos
Que se espalham n'amplidão
Parecem um bando de loucos
Com lanterninhas na mão,
Iluminando os caminhos
Sem saberem pr'onde vão

A resposta do modelo à entrada acima se encontra representada abaixo:

Para fazer a escansão e metrificação dos versos fornecidos,
→ primeiro vamos dividir cada verso em suas sílabas poéticas
→ e depois marcar as sílabas tônicas. Note que, na poesia, a
→ divisão silábica pode diferir da prosa devido a fenômenos
→ como a sinalefa (quando uma palavra termina em vogal e a
→ próxima começa com uma, elas se juntam em uma só sílaba
→ poética).

1. Centenas de pirilampos

- Cen-te-nas / de / pi-ri-lam-pos (8 sílabas poéticas)
 - Padrão rítmico: - u - u - u - u -
2. Que se espalham n'amplidão
 - Que / se / es-pa-lham / n'am-pli-dão (8 sílabas poéticas)
 - Padrão rítmico: - u - u - u - á
 3. Parecem um bando de loucos
 - Pa-re-cem / um / ban-do / de / lou-cos (8 sílabas poéticas)
 - Padrão rítmico: - u - u - u - u -
 4. Com lanterninhas na mão,
 - Com / lan-ter-ni-nhas / na / mão, (8 sílabas poéticas)
 - Padrão rítmico: - u - u - u - ão
 5. Iluminando os caminhos
 - I-lu-mi-nan-do / os / ca-mi-nhos (8 sílabas poéticas)
 - Padrão rítmico: - u - u - u - u -
 6. Sem saberem pr'onde vão
 - Sem / sa-be-rem / pr'on-de / vão (8 sílabas poéticas)
 - Padrão rítmico: - u - u - u - ão

A máquina não só tentou fazer a escansão, como também explicou sua linha de raciocínio – correta, diga-se de passagem. Mas mesmo com tamanha confiança, o resultado é visivelmente errôneo. Talvez seja de se esperar que um *large language model* tenha dificuldade em realizar uma tarefa ortográfica que tem raízes, de fato, na fonética, além de meandres amiúde nas regras.

Com a finalidade de adequar a resposta do modelo ao padrão adotado neste projeto, em sequência, adicionou-se ao final do *prompt* o seguinte adendo, referente à formatação:

O formato de saída desejado é o seguinte:

1. As sílabas são separadas pelo sinal gráfico " / ".
2. Cada sílaba tônica do verso é precedida pelo sinal gráfico
↪ " * ".
3. A última sílaba tônica deve ser sucedida pelo sinal gráfico
↪ " # ".

O resultado foi o abaixo:

Centenas de pirilampos:
Cen/te*nas/ de/ pi*ri/lam*pos#

Que se espalham n'amplidão:

Que/ se/ es*pal/ham/ n'a*mpli*dão#

Parecem um bando de loucos:

Pa*re/cem/ um/ ban*do/ de/ lou*cos#

Com lanterninhas na mão:

Com/ lan*ter/ni*nhas/ na/ mão#

Iluminando os caminhos:

I/lu*mi/nan*do/ os/ ca*mi/nhos#

Sem saberem pr'onde vão:

Sem/ sa*be/rem/ pr'on*de/ vão#

Note que a contagem de sílabas poéticas considera a sinalefa e

- a elisão (como em "n'amplidão" e "pr'onde"), e a tônica é
- marcada de acordo com a pronúncia natural das palavras,
- considerando também a regra da última sílaba tônica em
- versos paroxítonos terminados em uma vogal, "n", "m", ou
- ditongo oral, que é comum na poesia em língua portuguesa
- para determinar a métrica.

O modelo aparenta, a partir do *prompt* pura e simplesmente, ter entendido o modelo de formatação. Entretanto, ainda pecou ao separar as sílabas incorretamente, distribuir tônicas em locais impróprios e sempre finalizar com o sinal gráfico #, independentemente do término oficial da contagem de sílabas poéticas.

Já ao usar o modelo GPT-3.5-turbo-0125, as respostas foram ainda mais desencorajadoras. Ao se reciclar a mesma mensagem do *system* anterior, ele respondeu:

Centen*as / de pi*ri*lam*pos #
 Que se es*pal*ham / n'am*pli*dão #
 Pa*re*cem / um ban*do / de lou*cos #
 Com lan*ter*ni*nhas / na mão, #
 I*lu*mi*nando / os ca*mi*nhos #
 Sem sa*be*rem / pr'on*de vão #

Como se pode observar, o modelo devolveu uma resposta que separa apenas as palavras, e não as sílabas (isso quando não separa nem algumas palavras umas das outras), coloca tônicas em locais diversos e aleatórios e insere o término da contagem de sílabas poéticas sempre ao fim do verso, e não após a última tônica.

Ficou evidente que ambos os modelos não têm treinamento suficiente para realizar esta tarefa apenas com o poder do *prompt* limpo. O GPT-4, por ser um modelo mais avançado, naturalmente demonstrou mais destreza na realização. Na seção a seguir, outras técnicas serão utilizadas que tornam o *prompt* mais sofisticado.

4.0.2 Few-shot prompting

Few-shot prompting se difere do *zero-shot prompting* por trazer em seu corpo, além dos comandos do *prompt* um número reduzido de exemplos que "ensinam" rapidamente à máquina o tipo de resposta que se espera. Ou seja, o corpo da mensagem pode ser muito similar ao do *zero-shot prompt*, portanto que se proveja uma reduzida amostragem. O *prompt* escolhido foi o que segue, com temperatura também ajustada para 0.2

O texto escolhido para escansão desta vez foi a canção *Agalopado*, de Alceu Valença. Como o título sugere, sua estrutura é semelhante a um martelo agalopado. Este estilo comum dos cantadores do interior nordestino apresenta versos de 10 sílabas, com tônicas na 3^a, 6^a e 10^a. Os quatro primeiros versos do poema foram escandidos automaticamente pelo Aoidos e usados de exemplo para o *prompt*. Os 16 versos restantes (6 que sobraram da primeira estrofe e 10 da segunda estrofe) passaram pelo modelo GPT-4 através de chamada à API e foram escandidos um a um. Abaixo, o *prompt*:

Você é um excelente assistente de poeta.

Eu preciso fazer a escansão de alguns versos (escansão é a
 ↳ divisão silábica de versos poéticos e identificação das
 ↳ tônicas).

Quando eu colocar um poema como input, quero que você o
 ↳ devolva cada verso metrificado e com seu padrão rítmico
 ↳ assinalado. Ou seja, o verso deve ficar dividido em
 ↳ sílabas poéticas, com as tônicas dos versos marcadas.

O formato de saída desejado é o seguinte:

1. As sílabas são separadas pelo sinal gráfico " / ".
2. Cada sílaba tônica do verso é precedida pelo sinal gráfico
 ↳ " * ".
3. A última sílaba tônica deve ser sucedida pelo sinal gráfico
 ↳ " # ".

Exemplos:

USUÁRIO: Quando eu canto o seu coração se abala

RESPOSTA: Quando eu canto o seu coração se abala -> * Quan- /
 ↳ do eu / * can- / to o / * seu / co- / ra- / * ção / se a-
 ↳ / * ba- # / la

USUÁRIO: Pois eu sou porta-voz da incoerência

RESPOSTA: Pois eu sou porta-voz da incoerência -> Poi- / * s
 ↳ eu / sou / * por- / ta- / * voz / da in- / coe- / * rên- #
 ↳ / cia

USUÁRIO: Desprezando seu gesto de clemência

RESPOSTA: Desprezando seu gesto de clemência -> Des- / pre- /
 ↳ * zan- / do / seu / * ges- / to / de / cle- / * mên- # /
 ↳ cia

USUÁRIO: Sei que meu pensamento lhe atrapalha

RESPOSTA: Sei que meu pensamento lhe atrapalha -> * Sei / que
 ↳ / * meu / pen- / sa- / * men- / to / lhe a- / tra- / * pa-
 ↳ # / lha

USUÁRIO:

Como demonstra a primeira resposta da máquina (abaixo), as escansões saíram num formato que dispõe, primeiramente, o verso original; depois uma seta (->); e, em seguida, a separação propriamente dita. Mas, por conveniência, o resultado está visivelmente melhor formatado na Tabela 4.1.

Cego o sol seu cavalo de batalha -> * Ce- / go / o / * sol /
 ↳ seu / ca- / va- / * lo / de ba- / * ta- # / lha

Tabela 4.1 Versos escandidos com *few-shot prompting*

NUM	VERSO	ESCANSÃO
1	Cego o sol seu cavalo de batalha	* Ce- / go / o / * sol / seu / ca- / va- / * lo / de ba- / * ta- # / lha
2	E faço a lua brilhar no meio-dia	E / * fa- / ço a / * lu- / a / bri- / * lhar / no / * mei- / o- / * di- # / a
3	Tempestade eu transformo em calmaria	Tem- / * pes- / ta- / de eu / trans- / * for- / mo / em / cal- / * ma- # / ri- / a
4	E dou um beijo no fio da navalha	E / * dou / um / * bei- / jo / no / * fi- / o / da na- / * va- # / lha
5	Pra dançar e cair nas suas malhas	* Pra / dan- / çar / e / * cair / nas / * suas / * ma- # / lhas
6	Gargalhando e sorrindo de agonia	Gar- / ga- / * lhan- / do e / * sor- / rin- / do / de a- / * go- # / ni- / a
7	Se acaso eu chorar não se espante	* Se / a- / * ca- / so / eu / cho- / * rar / não / se es- / * pan- # / te
8	O meu riso e o meu choro não têm planos	O / * meu / * ri- / so / e / o / * meu / * cho- / ro / não / têm / * pla- # / nos
9	Eu canto a dor, o amor, o desengano	Eu * can- / to a / * dor# / , o / a- / * mor# / , o / de- / * sen- # / ga- / no
10	E a tristeza infinita dos amantes	E a / * tris- / te- / * za / in- / * fi- / ni- / * ta / dos a- / * man- # / tes
11	Don Quixote liberto de Cervantes	* Don / Qui- / * xo- / te / li- / * ber- / to / de / Cer- / * van- # / tes
12	Descobri que os moinhos são reais	Des- / co- / * bri / que / os / moi- / * nhos / são / re- / * ais #
13	Entre feras, corujas e chacais	En- / * tre / * fe- / ras, / co- / * ru- / jas / e / cha- / * cais#
14	Viro pedra no meio do caminho	* Vi- / ro / * pe- / dra / no / * mei- / o / do / ca- / * mi- # / nho
15	Viro rosa, vereda de espinhos	* Vi- / ro / * ro- # / sa, / ve- / * re- / da / de / es- / * pi- # / nhos
16	Incendeio esses tempos glaciais	In- / * cen- / de- / i# / o / es- / * ses / * tem- / pos / gla- / * ci- # / ais

Visivelmente, a performance melhorou. Boa parte das metrificação respeita pelo menos as regras ortográficas de separação. Em algumas delas, a sinalefa é corretamente aplicada ("/ ço a /", verso 2). Entretanto, em casos como na linha 4 ("/ da na- /"), junta-se duas sílabas onde não é possível sinalefa. As acentuações, no entanto, parecem estar distribuídas de maneira aleatória, e nem sempre casam com as tônicas das palavras. Já o sinal de término (#), em várias partes de fato termina na última tônica; em outras, não casa com a última tônica; e em três casos (9, 14, 16) surgem mais de uma vez na escansão. São muitas regras, muitos casos específicos, muitas excessões; faz-se necessário ainda um pouco mais de granulação.

4.0.3 Chain of thought

Como há um caráter algorítmico na escansão poética, em que se aplica-se as tônicas, identifica-se as sílabas do verso, faz-se a divisão e se define o fim do verso, *Chain of Thought* (CoT) se torna uma candidata possível a realizar este processo automaticamente. Há duas formas de se aplicar CoT: ou como uma técnica de *few-shot prompting*, ou *zero-shot prompting*. Há indícios de que primeira forma fornece uma melhora substancial em respostas de LLMs para questões aritméticas que envolviam vários passos.

Benchmarks de aritmética passo-a-passo e raciocínio lógico têm desafiado os limites da escalabilidade de *large language models* [Rae et al., 2021]. *Chain of thought* (CoT) *prompting* [Wei et al., 2022], uma instância do *few-shot prompting*, propõe uma solução simples, ao modificar as respostas, de poucos exemplos para respostas passo-a-passo, alcançando melhoria significativa de performance.(KOJIMA et al., 2023, p. 3, tradução nossa)

Nesta primeira técnica, fornece-se um exemplo análogo de como resolver a tarefa passo-a-passo e, em seguida, exhibe-se o problema que quer resolver. Por exemplo, se temos o seguinte problema: "Um malabarista consegue manipular 16 bolas simultaneamente. Metade das bolas são de golfe, e metade das bolas de golfe são azuis. Quantas bolas de golfe azuis tem o malabarista?" Com o *few shot CoT*, pode-se no início do *prompt* oferecer um exemplo de par pergunta/resposta e, em seguida, a pergunta que se quer de fato responder:

PERGUNTA: Roger tem cinco bolas de tênis. Ele compra mais duas
→ latas de bola de tênis. Cada lata tem 3 bolas. Quantas
→ bolas ele tem agora?

RESPOSTA: Roger começou com 5 bolas. 2 latas de 3 bolas têm 6
→ bolas no total. $5 + 6 = 11$. A resposta é 11.

PERGUNTA: Um malabarista consegue manipular 16 bolas
→ simultaneamente. Metade das bolas são de golfe, e metade
→ das bolas de golfe são azuis. Quantas bolas de golfe azul
→ tem o malabarista?

RESPOSTA:

Já um *zero-shot CoT* não fornece exemplos, apenas uma frase-chave que "estimule" a máquina a pensar sistematicamente, algo na linha "Vamos raciocinar passo-a-passo". Um estudo recente (KOJIMA et al., 2023) demonstrou que LLMs podem ser "*zero-shot reasoners*", ou seja, capazes de melhorar o pensamento lógico apenas com *prompts* estruturados para raciocínios específicos. No experimento conduzido pelo presente trabalho, usamos o GPT-4-turbo-preview com um *zero-shot CoT*, explicando um passo-a-passo de como fazer uma escansão, junto com algumas das regras comumente aplicadas na metrificação.

Em seu *Tratado de versificação*, Olavo Bilac expõe regras para a separação em sílabas poéticas que considera "princípios indispensáveis que não se devem absolutamente desprezar nem sequer deixar de ter presentes (sic)"(BILAC; PASSOS, 1905, p. 38) Algo longo de algumas

páginas, o parnasiano foca principalmente nos fenômenos da sinalefa e da sinérese, apontando-os como os mais comuns para a escansão poética. A diérese nem figura como metaplasmo no livro; o que o Bilac faz, no máximo, é uma distinção de determinadas pronúncias que mudaram com o tempo: "Camões contava (como outros antigos) em saudade 4 syllabas [...]; isto, porém, ha muito caiu em desuso (sic)"(BILAC; PASSOS, 1905, p. 39)

Tendo em mãos o tratado de Bilac, a dissertação de Adiel Mittmann e outras noções de versificação extraídas arbitrariamente dos livros *Versos, sons e ritmos* (GOLDSTEIN, 2005), *Análise do poema* (GOLDSTEIN, 1988) e *O estudo analítico do poema* (CANDIDO, 2006), foi construído um *prompt* que generaliza essas regras para uma abordagem *chain of thought*. Abaixo, o que foi construído:

Você é um excelente assistente de poeta.

Eu preciso fazer a escansão de alguns versos (escansão é a
→ divisão silábica de versos poéticos e identificação das
→ tônicas).

Quando eu colocar um poema como input, quero que você o
→ devolva cada verso metrificado e com seu padrão rítmico
→ assinalado. Ou seja, o verso deve ficar dividido em
→ sílabas poéticas, com as tônicas dos versos marcadas. Siga
→ o roteiro abaixo:

1. Identifique os versos. Cada verso é separado por uma quebra
→ de linha.
2. Antes de fazer a divisão em sílabas poéticas, identifique
→ quais são as sílabas tônicas de cada palavra.
3. Procure por sinalefas e sinéreses:
 - a) Se o final de uma palavra terminar em vogal e o início da
→ palavra seguinte também, temos uma sinalefa e, portanto,
→ fazem parte da mesma sílaba poética.
 - b) Para fazer sinéreses: hiatos ficam juntos em uma só sílaba,
→ assim como ditongos.
4. Agora, pode efetivamente fazer a divisão em sílabas
→ poéticas. Use o sinal gráfico "/" para fazer a divisão
→ entre elas, no formato "sílaba / sílaba / sílaba /...".
5. Após fazer a divisão, resgate as sílabas que identificou
→ como tônicas. Vá nas sílabas que lhes correspondem e
→ marque-as com o sinal gráfico "*" antes da sílaba que é
→ tônica, no formato "/ * sílaba").
6. Por último, identifique a última tônica demarcada, ou seja,
→ o último sinal "*". Use o sinal gráfico "#" para demarcar
→ essa última tônica. Coloque o "#" logo após a sílaba, de
→ forma que fique no formato "/ * sílaba #".

- a) Só pode haver um "#" por verso, exclusivamente na última
 ↳ tônica.

Reforçando o formato de saída desejado:

1. As sílabas são separadas pelo sinal gráfico " / ".
2. Cada sílaba tônica do verso é precedida pelo sinal gráfico
 ↳ " * ".
3. A última sílaba tônica deve ser sucedida pelo sinal gráfico
 ↳ " # ".

Vamos raciocinar passo-a-passo.

No entanto, a resposta dada pelo GPT-4 não foi satisfatória:

Para realizar a escansão, seguiremos o passo a passo indicado,
 ↳ atendendo todas as solicitações e normas de divisão em
 ↳ sílabas poéticas. Vamos começar:

1. **Do Gosto para o desgosto**
 - Do / *Gos# / to / pa / *ra / o / des / *gos / to
2. **O quadro é bem diferente**
 - O / *quad# / ro / é / bem / di / *fe / ren / te
3. **Ser moço é ser sol nascente**
 - Ser / *mo# / ço / é / ser / *sol / nas *cen / te
4. **Ser velho é ser um sol posto**
 - Ser / *vel# / ho / é / ser / um / *sol / pos / to
5. **Pela rugas do meu rosto**
 - Pe / *la / ru #/ gas / do / me / u / *ros / to
6. **O que fui hoje não sou**
 - O / que / *fu# / i / *ho / je / não / *sou
7. **Ontem estive hoje não estou.**
 - On / tem / es / *ti# / ve / *ho / je / não / es / tou.
8. **Que o sol ao nascer fulgura**
 - Que / o / *sol / ao / nas / *cer / ful #/ gu / ra
9. **A parte que iluminou**
 - A / *par# / te / que / i / lu / mi / *nou

Lembrando que as marcas de sílaba tônica são representadas
→ pelo " * " antes da sílaba e o último acentoônico do
→ verso é seguido por " # ".

Mesmo com o pedido explícito (e a lembrança do pedido na resposta da própria máquina), na maioria das vezes o modelo não respeitou a regra de colocar o sinal "#" exclusivamente após a última tônica. Mesmo com regras simplificadas de sinalefa e sinérese (evitando falar, e.g., de vogais fortes e fracas), nenhum encontro vocálico ficou junto. Pelo contrário, na sexta linha, o verbo "fui" foi separado em "fu / i". Já as tônicas estão, em diversos versos, mal distribuídas.

4.0.4 Fine-tuning

4.0.4.1 Por quê fine-tuning?

Ao se esgotar as opções de *prompt engineering*, um desenvolvedor trabalhando com LLMs se depara com duas alternativas para a resolução do seu problema, ou vai com *Retrieved Augmented Generation* (RAG) ou pega o bonde do *fine-tuning*. A decisão de aplicar uma ou outra depende da situação.

Se a máquina não atinge o esperado de *output* porque o modelo não tem acesso a um conhecimento necessário, então faz-se necessário otimizar o contexto, ou seja, fornecer a informação faltante. Por outro lado, se o modelo não estiver correspondendo às expectativas de estilo ou formato, a otimização deve ser feita na própria LLM. (ALLARD; JARVIS, 2023)

A Figura 4.1 representa bem esse fluxo. Nela, toda solução parte primeiramente de uma tentativa de *prompt engineering*. Caso o procedimento falhe, RAG é apontada como possível solução para otimizar o contexto, enquanto *fine-tuning* aparece como uma estratégia para otimizar o comportamento da máquina.

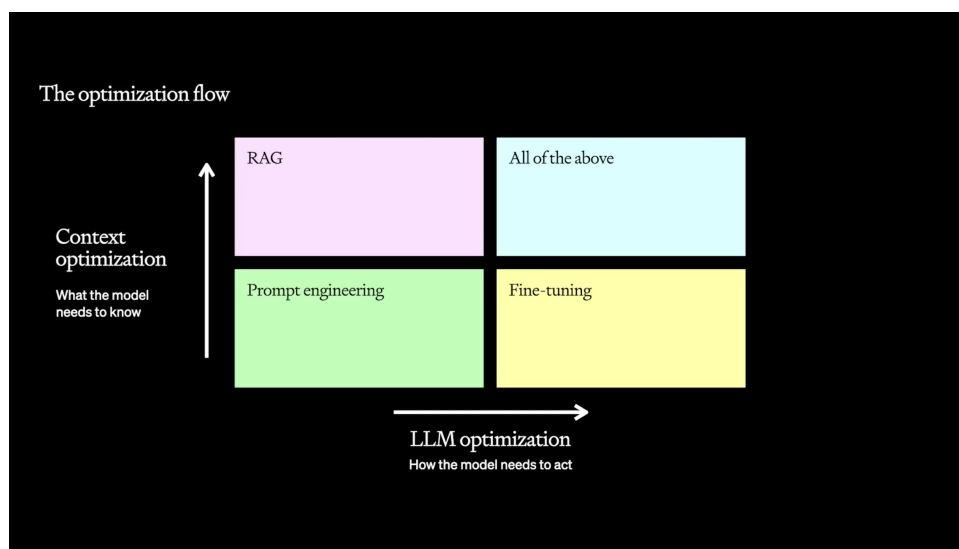


Figura 4.1 Gráfico demonstra o fluxo de otimização. Parte-se de *prompt engineering*; sobe-se pelo eixo y se o intuito é melhorar o contexto; anda-se pelo eixo x caso seja necessário aprimorar o comportamento da LLM; chega-se à ponta superior direita caso haja essa dupla necessidade. (ALLARD; JARVIS, 2023)

De acordo com a documentação da própria OpenAI, os modelos disponíveis na plataforma são treinados para serem assistentes de propósito geral, mas o *fine-tuning* pode afunilar o proceder do modelo, e fazê-lo adquirir padrões de comportamento específicos.

Já dá para perceber o resumo da ópera. No caso do presente projeto, em que o intuito não é recuperar informações novas, e sim condicionar o *output* do modelo a respeitar um determinado padrão de separação silábica, *fine-tuning* se torna o próximo passo lógico para a pesquisa.

4.0.4.2 Conjunto de dados

Para começar o processo de *fine-tuning*, é necessário, primeiramente, coletar dados estruturados com um número representativo de exemplos. A documentação da OpenAI explica que, para se treinar uma máquina, é preciso ter pelo menos 10 exemplos, mas que, para se enxergar as mudanças de maneira mais efetiva, deve-se inserir pelo menos 50 exemplos. É neste momento que entra o Aoidos, um sistema que gera, automaticamente, escansões para poemas.

No entanto, como foi esmiuçado no capítulo 3 deste trabalho, o Aoidos faz a escansão dos versos de maneira contextual. Ele considera a estrofe como um todo e gera escansões que almejam adequar cada verso a uma métrica e um esquema rítmico consistente, aplicando regras de metaplasmos para atingir este resultado.

O sistema Aoidos foi testado num *corpus* gigantesco para garantir sua proficiência. Entretanto, esse *corpus* foi entendido como poemas completos, com os núcleos sendo as estrofes:

[F]oram selecionadas obras suficientes para que fossem atingidos no mínimo 100.000 versos. [...] As obras cobrem mais de três séculos de poesia, do prestigioso épico de Camões aos poemas da catarinense Delminda Silveira. (MITTMANN, 2016, p. 57)

O nosso objetivo é outro. A ideia aqui é que um verso, singularmente, seja ingressado no modelo como *input* e tenha uma escansão válida devolvida pelo *output*, de preferência, na sua forma mais "natural".

A maneira que foi encontrada de padronizar estes dados foi usar de um fragmento do *corpus* usado como teste no projeto do Aoidos¹, contendo cerca de 13.500 versos.

Ao coletar todos esses versos, criamos um *script* em Python para separá-los linha a linha, como se cada linha fosse uma estrofe em si mesma. Esses dados então foram lançados em arquivos txt contendo 800 versos separados por duas quebras de parágrafo. A plataforma *online* do Aoidos não consegue processar um volume de 13.500 versos de uma vez, portanto, empiricamente, esse número foi sendo reduzido até se encontrar o valor aproximado de 800 versos por vez que não quebraria o sistema. Os arquivos txt são convertidos pelo Aoidos para um xml com certas informações para o sistema e, em seguida, levados ao processamento.

O resultado foram 17 *outputs*, cada qual com 800 versos (exceto o último, com 634), totalizando 13.434 versos. A Figura 4.2 revela a escansão de alguns desses quase 13.500 versos escandidos automaticamente pelo Aoidos.

Isometric group of verses, each with 7 syllables

3-7 | do meu tris- te co- ra- ção!

Isometric group of verses, each with 7 syllables

3-7 | O tra- ba- lho no- bi- li- ta.

Isometric group of verses, each with 7 syllables

2-4-7 | A- mor su- bli- me pal- pi- ta

Isometric group of verses, each with 6 syllables

3-6 | do O- pe- rá- rio no pei- to

Isometric group of verses, each with 7 syllables

4-7 | que lhe en- si- na es- te pre- cei- to:

Figura 4.2 Alguns dos versos escandidos pelo Aoidos

¹Repositório com excerto dos conjunto de dados usado no teste do sistema Aoidos: <https://github.com/adiel-mittmann/poemas>

Essa abordagem garante que, quando os dados são lançados para *fine-tuning*, a máquina não fique "confusa", ao ser-lhe apresentada muitas maneiras contraditórias de se escandir um verso. A palavra "saudade" -- como já foi levantado algumas vezes aqui -- ou pode ganhar uma diérese no ditongo "sau", sendo metrificada como "sa / u / da / de", ou pode ser metrificada mais comumente como "sau / da / de". No nosso caso, inconsistências como essas devem ser mitigadas², para que os dados sejam aproveitados de maneira mais consistente.

Por último, criamos um outro *script* em Python, utilizando a biblioteca *BeautifulSoup*, que extrai o código HTML da página, identifica os versos através das tags, processa-os para atingirem uma padronização desejada e guarda-os em um arquivo csv, junto com alguns metadados (esquema rítmico e classificação do verso -- se é tetrassílabo, heroico, sáfico etc.). Um exemplo de linha do arquivo csv aparece na Tabela 4.2.

Tabela 4.2 Excerto do csv com dados coletados do Aoidos

verse	metrical_classification	rhythmic_pattern	syllables_stressed
nos verdes ramos	tetrassílabo	2-4	nos / * ver- / des / * ra- # / mos

Realizado todo esse processo, coletou-se um volume robusto de exemplos de treino, com o maior grau de consistência possível nos *outputs*. As escansões produzidas desta forma revelam versos com sua métrica e esquema rítmico mais desnudos de intervenções dos metaplasmos que desnaturalizam a oralidade. Um poeta popular que se valesse de versos tão consistentes, diria-se que, dos três pilares que deve dominar (rima, métrica e oração), certamente teria aptidão para a métrica.

4.0.4.3 Treinamento

Numa primeira abordagem, com o intuito de observar os custos de treinamento e tentar encontrar um teto de performance, treinamos os dados de maneira iterativa. A OpenAI permite, para alguns modelos, incluindo o GPT-3.5-turbo-0125, treinar modelos que já foram pré-treinados. I.e., é possível treinar um modelo com um conjunto de dados, salvá-lo e, em seguida, usar outro conjunto de dados para treiná-lo novamente. Esta *feature* é importante para quando se trabalha num processo iterativo, como no caso de treinar um *chatbot* com modelos de conversa e, à medida que novas conversas (reais ou sintéticas) são produzidas, permite-se refinar a máquina.

Fizemos os testes com um valor de 30% do total de exemplos da base de treinamento. Portanto, na primeira iteração, treinamos 400 exemplos e testamos e validamos 120 exemplos que não faziam parte da base de treinamento. Para a segunda iteração, treinamos 400 exemplos em cima do modelo resultante da primeira iteração, testamos e validamos 240 exemplos. Para a terceira, com mais 400, treinamos e validamos 360 exemplos. Isto ocorreu sucessivamente até décima oitava -- e última -- iteração, com um total acumulado de 7200 exemplos treinados e 2160 exemplos de teste.

²Ainda sim, surgem determinadas particularidades contextuais do verso em si, ou dos critérios de naturalidade aplicados pelo sistema, que divergem de uma padronização total. Todavia, a heurística aplicada serviu para amainá-las.

Numa segunda experimentação, treinou-se uma quantidade de dados mais robusta de uma só vez no GPT-3.5-turbo-0125 (limpo, sem estar pré-treinado). Foram 3520 exemplos oferecidos para a máquina, com um conjunto de testes de 880 exemplos.

O *fine-tuning* do GPT-3.5-turbo-0125 recebe como formatação um arquivo JSONL com três campos a serem preenchidos. Um deles é a mensagem para o *system*, que determina o comportamento do assistente. O outro é um campo para o *user*, que representa um possível *input* do usuário. O último campo é o *assistant*, que determina a saída esperada para aquele *input*. Uma formatação em JSON para apenas um exemplo ficaria assim:

```

1  { "messages" : [ {
2    "role": "system",
3    "content" : "Você é um algoritmo para dividir versos em
4      ↳ sílabas poéticas. O usuário vai te enviar como input
5      ↳ um verso e você devolverá a ele sua divisão em
6      ↳ sílabas poéticas. Por exemplo:
7
8    Usuário: Teus formosos quinze anos
9    Assistente: Teus formosos quinze anos -> Teus / for- / *
10     ↳ mo- / sos / quin- / * ze a- # / nos
11
12    O sinal gráfico / sinaliza a divisão da sílaba poética.
13    O sinal gráfico * sinaliza as tônicas do verso.
14    O sinal gráfico # identifica o final da contagem das
15     ↳ sílabas, pois sílabas poéticas só vão até a última
16     ↳ tônica." },
17
18    { "role": "user",
19      "content": "Não há dos mares à flor!" },
20
21    { "role": "assistant",
22      "content": "Não há dos mares à flor! -> Não h- / * á /
23     ↳ dos / * ma- / re- / s à / * flor! #" } ] }
```

Quanto à seleção de hiperparâmetros para treinamento, seguiu-se a diretriz da própria OpenAI³, que recomenda

³<https://platform.openai.com/docs/guides/fine-tuning/use-a-fine-tuned-model>

inicialmente treinar sem especificar nenhum desses [hiperparâmetros], permitindo-nos determinar um padrão baseado no tamanho do seu *dataset*, em seguida ajuste-os se perceber o seguinte:

- * Se o modelo não se adequa ao conjunto de treinamento tanto quanto esperado, aumente o número de épocas em 1 ou 2 [...]
- * Se o modelo se tornar menos diverso do que era esperado, diminua o número de épocas em 1 ou 2 [...]
- * Se o modelo não parece estar convergindo, aumente o *rate multiplier* [...]

(tradução nossa)

Como o treinamento apresentou resultados aceitáveis a cada passagem, mesmo com alguns momentos de perda de eficiência de acordo com nossas métricas (descritas na seção abaixo), optou-se por manter os hiperparâmetros de treinamento *default*, a cargo da própria plataforma (inclusive por uma questão de consistência). Não aparenta ter havido *overfitting* ou fuga da expectativa de *output*.

Para a geração de respostas, em que se insere o verso e se espera o *output* do modelo treinado, o único hiperparâmetro modificado foi a temperatura, ajustada para 0.1, que espera respostas mais determinísticas do modelo, evitando o risco de alucinação da máquina, ou de uma demonstração de "criatividade" que não lhe compete neste determinado contexto.

4.0.4.4 Performance

Há inerentes obstáculos na tentativa de se avaliar a qualidade do resultado de uma LLM, principalmente quando a subjetividade se impõe de maneira mais aguda. No experimento do PoeLM, sistema que gera poemas estruturados em espanhol e em basco, os pesquisadores tiveram que recorrer a critérios de avaliação que envolviam uma enquete com usuários para determinar não só a estrutura, mas qualidade do que foi gerado. "Avaliadores humanos preferem os poemas gerados pelo nosso sistema aos escritos por poetas renomados em 37.3% dos casos". (ORMAZABAL et al., 2022, p. 5, tradução nossa)

Para sorte de um escansor pura e simplesmente, há critérios mais objetivos capazes de dar uma visão da sua eficiência. A primeira intuição é fazer uma comparação de strings entre a escansão do Aoidos e o texto gerado pelo LLM. A Tabela 4.3 reflete esta métrica. O segundo modelo iterado teria tido a pior performance, e o último, a melhor. O modelo "single" se refere ao testado de uma só vez, com 3520 exemplos.

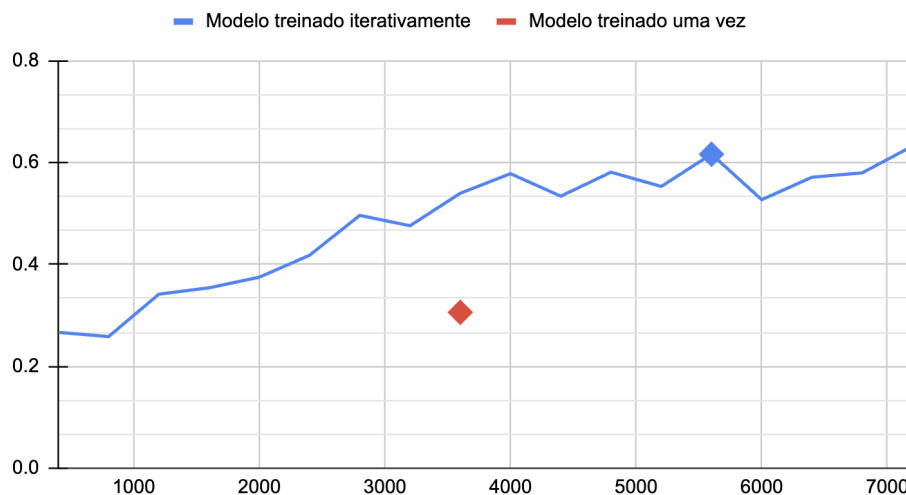
Há uma clara evolução, apresentada na Figura 4.3, mas perde-se muito da granularidade do que a máquina estaria devolvendo. E se há pequenos problemas de formatação? Ou se ele acerta boa parte da escansão e depois se perde um pouco? E não acerta todas as sílabas, mas acerta as quantidades? Por essas razões, outras três métricas foram desenhadas para tentar entender ainda melhor a evolução do modelo, a saber:

1. SYL_MATCH_% - Compara-se sílaba a sílaba entre a escansão feita pelo Aoidos e a escansão feita pelo modelo no conjunto de teste. Isto gera um percentual de acerto, que é somado e dividido pela quantidade de exemplos no conjunto de teste, dando um percentual geral do *match* de sílabas;

Tabela 4.3 Match total de strings entre o verso escandido por Aoidos e o gerado por LLM

MODEL	TRAIN	TEST	TOTAL_MATCH_%
1	400	120	26.67%
2	800	240	25.83%
3	1200	360	34.17%
4	1600	480	35.42%
5	2000	600	37.50%
6	2400	720	41.81%
7	2800	840	49.64%
8	3200	960	47.60%
9	3600	1080	53.98%
10	4000	1200	57.83%
11	4400	1320	53.41%
12	4800	1440	58.13%
13	5200	1560	55.32%
14	5600	1680	61.67%
15	6000	1800	52.72%
16	6400	1920	57.14%
17	6800	2040	57.99%
18	7200	2160	63.24%
SINGLE	3520	1760	0.30625

2. SYL_MATCH_UN\$_% - Opera igual ao SYL_MATCH_%, mas com um detalhe: antes de se fazer a comparação, remove-se os marcadores de tônica (), via um *script* em Python.
3. SYL_COUNT_MATCH_% - Conta-se a quantidade de sílabas poéticas (excluindo-se as que ficam após do término, #) entre a escansão feita pelo Aoidos e a escansão feita pelo modelo no conjunto de teste. Compara-se se houve acerto da quantidade de sílabas e se divide a quantidade de acertos pelo total de exemplos no conjunto de testes.

TOTAL_MATCH_%**Figura 4.3** SYL_MATCH_%. Os losangos revelam os melhores resultados de cada abordagem.

Para entender melhor esses novos critérios, observa-se primeiramente a tabela 4.4. Ela demonstra, na primeira linha, o verso gerado pelo Aoidos e, na segunda, o verso gerado pelo modelo treinado (primeira iteração de 400 exemplos) para o verso "Verás, sereno, altivo, triunfante,". A comparação se dá sílaba a sílaba. Na quarta sílaba, os versos deixam de se equivaler, e sílabas como "vo,", mesmo que iguais, já não contam como um *match*. Vale salientar que o verso do modelo ainda é uma escansão válida, apresentando dialefa no encontro entre "sereno" e "altivo", e diérese no hiato de "triunfante". Ainda sim, recebe uma pontuação baixa em similaridade de sílabas, e não dá *match* na quantidade de sílabas, pontuando 0 neste aspecto.

Comparação entre verso do Aoidos e gerado pelo modelo (com tônica)										
Ve-	* rás,	se-	* re-	no, al-	* ti-	vo,	triun-	* fan- #	te,	
Ve-	* rás,	se-	* re-	no,	al-	* ti-	vo,	tri-	un-	* fan- # te,

Tabela 4.4 SYL_MATCH_% = 44,4%, SYL_COUNT_MATCH_% = 0 (quantidade de sílabas Aoidos = 9, quantidade de sílabas do modelo = 11)

Para o verso "Nem ouvir, nem falar apetecera:", temos as duas escansões da tabela 4.5. A primeira é do Aoidos; a segunda, do modelo iterativo treinado com um total de 2800 exemplos. Percebe-se que a comparação tem a pontuação zerada, tanto em similaridade de sílabas quanto em quantidade de sílabas. Isto quer dizer que não há um *match* sílaba a sílaba, nem muito menos o modelo adivinhou a quantidade de sílabas proposta pelo Aoidos.

Comparação entre verso do Aoidos e gerado pelo modelo (com tônica)										
* Nem	ou-	vir,	* nem	fa-	* la-	r a-	pe-	te-	* ce- #	ra:
Ne-	m ou-	* vir,	ne-	m fa-	* lar a-	pe-	te-	* ce- #	ra:	

Tabela 4.5 SYL_MATCH_% = 0%, SYL_COUNT_MATCH_% = 0 (quantidade de sílabas Aoidos = 10, quantidade de sílabas do modelo = 9)

Ao remover-se os marcadores de tônica das duas escansões do verso anterior, há uma mudança de resultado. Agora, a terceira sílaba ("vir") dá *match* nas duas escansões. O percentual de similaridade do verso sem as tônicas sobe para 10%. Um acerto apenas de divisão silábica pode ser útil, pois sístoles e diástoles são muito comumente aplicadas na poesia popular como forma de forçar uma métrica a um ritmo. Portanto, um verso com a metrficação correta pode ter suas tônicas levemente modificadas na oralidade.

Comparação entre verso do Aoidos e gerado pelo modelo (sem tônica)										
Nem	ou-	vir,	nem	fa-	la-	r a-	pe-	te-	ce-	ra:
Ne-	m ou-	vir,	ne-	m fa-	lar a-	pe-	te-	ce-	ra:	

Tabela 4.6 SYL_MATCH_UN_% = 10%.

Uma observação: no 14^a treinamento do modelo, este último verso ("Nem ouvir, nem falar apetecera:") figurou com 100% de *match* silábico, demonstrando evolução no sistema.

Explicadas as métricas traçadas para o modelo, a Tabela 4.7 demonstra os valores acumulados de cada uma delas, iteração a iteração, nos modelos treinados com acumulação progressiva de exemplos.

Tabela 4.7 Métricas de um mesmo modelo treinado iterativamente

ITERATION	TRAIN	TEST	SYL_MATCH_%	SYL_MATCH_UN_%	SYL_COUNT_MATCH_%
1	400	120	70.15%	76.14%	67.50%
2	800	240	71.29%	78.94%	73.33%
3	1200	360	74.15%	78.84%	74.72%
4	1600	480	75.85%	82.44%	77.92%
5	2000	600	79.32%	84.82%	73.83%
6	2400	720	77.66%	82.36%	75.69%
7	2800	840	81.42%	87.00%	81.55%
8	3200	960	82.03%	86.44%	79.48%
9	3600	1080	85.86%	90.38%	84.35%
10	4000	1200	86.26%	89.81%	84.75%
11	4400	1320	83.46%	87.88%	82.42%
12	4800	1440	86.01%	89.61%	84.51%
13	5200	1560	86.49%	91.20%	85.51%
14	5600	1680	88.60%	92.30%	87.56%
15	6000	1800	83.58%	87.95%	81.94%
16	6400	1920	85.39%	91.57%	86.46%
17	6800	2040	85.95%	90.80%	85.98%
18	7200	2160	86.88%	90.12%	85.32%

Como é possível notar pelo campo assinalado, a 14ª iteração obteve os melhores resultados em todos os campos. Ficará mais claro nos gráficos plotados à frente que há uma tendência de crescimento praticamente a cada iteração, com "vales" de perda de eficiência distribuídos ao longo do processo.

De maneira comparativa, a Tabela 4.9 apresenta um resultado muito aproximado aos treinados iterativamente, quando se leva em conta o volume do conjunto de treinamento. Até mais alto.

Tabela 4.8 Métricas dos modelos treinados unitariamente

MODEL	TRAIN	TEST	SYL_MATCH_%	SYL_MATCH_UN_%	SYL_COUNT_MATCH_%
1	3520	880	87.19%	91.45%	88.30%

Tabela 4.9 Métricas do modelo treinado de uma vez só.

Para se entender melhor o impacto causado pelo volume de exemplos de treino, é mister observar as Figuras 4.4, 4.5 e 4.6. A Figura 4.4 mostra a evolução do critério SYL_MATCH_%. A linha azul acompanha o movimento dos modelos treinados iterativamente; o ponto vermelho mostra o modelo treinado de uma só vez. Embora tenha havido uma queda repentina na 15ª iteração, há um movimento de recuperação similar aos ocorridos da 5ª para a 7ª iterações e da 10ª para a 12ª iterações, mesmo que mais acentuado. Uma interpretação possível é de que ainda podemos manter a tendência de crescimento, dado um número maior de exemplos.

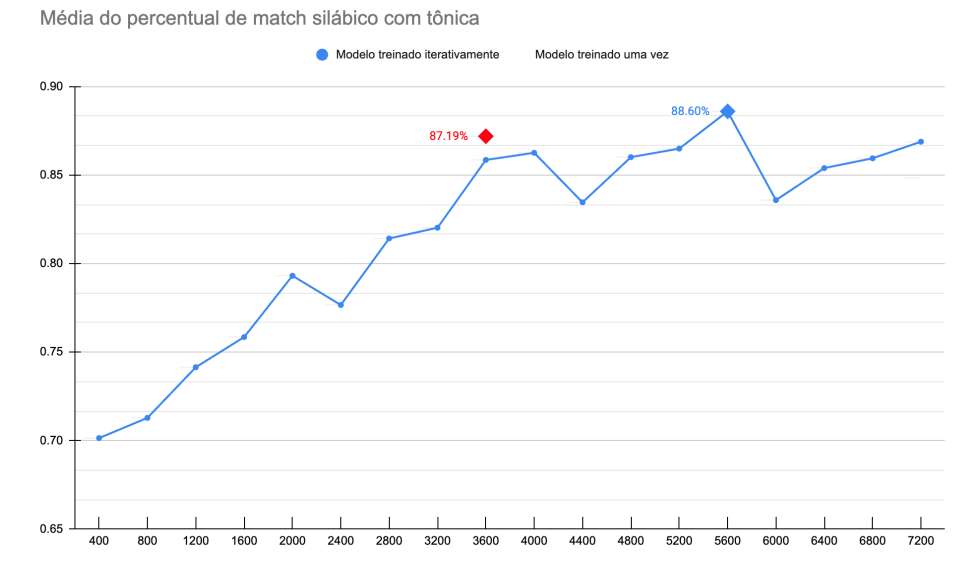


Figura 4.4 SYL_MATCH_%. Os losangos revelam os melhores resultados de cada abordagem.

Para a Figura 4.5, o movimento dos modelos iterativos para o critério SYL_MATCH_UN_% é muito similar ao anterior. Há uma oscilação de performance no decorrer das passagens, mas uma trajetória média ascendente. Assim como no gráfico anterior, o ponto vermelho mostra valor acima para quantidade comparável de exemplos.



Figura 4.5 SYL_MATCH_UN_%. Os losangos revelam os melhores resultados de cada abordagem.

Por fim, a Figura 4.6 demonstra um serpenteio parecido com os anteriores para o critério de SYL_COUNT_MATCH_%. Isto indica que essas métricas estão fortemente correlacionadas, e

que o treinamento sucessivo do modelo com mais exemplos é capaz de melhorar a performance, com alguns pontos de depressão ao longo do caminho.



Figura 4.6 SYL_COUNT_MATCH_%. Os losangos revelam os melhores resultados de cada abordagem.

Tendo em vista que metaplasmos que "desnaturalizam" um poema são frequentemente aplicados por poetas – e de que ligeirezas e vagariedades podem ser aplicadas à enunciação dos versos na pronúncia oral –, uma última métrica foi gerada, apenas para o modelo resultante da 14ª iteração (com melhor performance). Criando-se uma margem de erro de 1, para mais ou para menos, reprocessou-se o SYL_COUNT_MATCH_%. Desta vez, por exemplo, um verso que apresentasse 10 sílabas no Aoidos poderia apresentar entre 9 e 11 na escansão por LLM. O resultado após esse "jeitinho" de contar o acerto de quantidade de sílabas poéticas está descrito na tabela 4.10.

Tabela 4.10 Métrica com margem de erro para modelo 14

SYL_COUNT_MATCH	TEST	SYL_COUNT_MATCH_%
1636	1680	97.38%

Tabela 4.11 Métrica para *match* de número de sílabas com margem de erro de 1, para mais ou para menos

Isto é, mais de 97% das vezes, o escansor fez a separação silábica no entorno da quantidade de sílabas de um verso escandido, segundo o Aoidos, da forma fonética mais "natural".

Model	Input	Output	per
gpt-3.5-turbo-0125	\$0.0005	\$0.0015	1K tokens

Tabela 4.12 Disponível em: <https://openai.com/pricing>

4.1 Custo

Deve-se ter notado que o máximo de dados utilizado no treinamento dos modelos foi inferior à quantidade coletada com o processamento do Aoidos. Enquanto cerca de 13.500 versos foram escandidos automaticamente pelo sistema, apenas 7.200 acabaram como exemplos de *fine-tuning*. Isto se deu porque houve mais um critério foi adotado para a viabilidade do projeto: *fine-tuning* é uma abordagem que exige um planejamento prévio antes de ser executada, porque ela é *sensível ao custo*.

Navegar pelo *dashboard* e a documentação da OpenAI em busca objetiva de como se manifestam os custos pode ser uma experiência kafkaniana, ou seja processo pode ser absurdamente complicado, opressivamente burocrático, confuso ou irracional. O *dashboard* de usos fornece mais explicitamente apenas um valor acumulado de suas atividades. A página específica de precificação na documentação distingue dois valores, de *input* e de *output*, como demonstrado na tabela 4.12.

Entretanto, não fica claro o que seriam os *inputs* e *outputs*. Os *inputs* seriam apenas as chamadas à api com os *prompts*, ou o *input* de dados de treinamento também contariam aqui? Numa investigação mais aprofundada, a página de "*Preparing your dataset*"⁴ manifesta uma explicação sobre custos de treinamento do GPT-3.5 ligeiramente mais precisa:

Para estimar o custo de um trabalho de *fine-tuning*, use a fórmula a seguir:

custo base de 1k *tokens* * número de *tokens* no arquivo de entrada * número de épocas treinada

Para um arquivo de treinamento com 100.000 *tokens* treinados em 3 épocas, o valor esperado custaria em torno de \$2.40 USD.

(tradução nossa)

Fazendo uma engenharia reversa e uma conta de padaria, se o custo para treinar 100.000 *tokens* por 3 épocas é US\$2,40, então o custo de se treinar 1 *token* é de US\$0,000008. Todos os modelos do presente projeto foram treinados por 3 épocas.

Sendo US\$0,0000005 para processar um tokens de *input* e US\$0,0000015 a cada token de resposta, cada chamada à API retornava um callback com o consumo de tokens. A partir daí, o valor por treinamento foi calculado nas tabelas 4.15 e 4.18. Em geral, uma chamada à API para fazer a escansão gasta em torno de 550 *tokens* (juntando os *tokens* do *prompt* e os *tokens* do *input*), como demonstrado na Tabela 4.14, que apresenta o *callback* de um verso da música *Deixa acontecer* do grupo Revelação, escandido pelo 14º modelo.

⁴<https://platform.openai.com/docs/guides/fine-tuning/common-use-cases>

Tabela 4.13 Custo dos modelos treinados iterativamente

GENERATED	PROMPT_T	COMP_T
dei- / xa a- / con- / te- / ce- / r / na- / tu- / ral- / men- # / te	505	43

Tabela 4.14 GENERATED: verso escandido pelo LLM; PROMPT_T: quantidade de tokens de input; COMP_T: quantidade de tokens da resposta do LLM

Tabela 4.15 Custo dos modelos treinados iterativamente

MODEL	TRAIN_T	TRAIN_COST	PROMPT_T	PROMPT_COST	COMP_T	COMP_COST	TOTAL_US
1	254193	6.100632	61200	0.03060	5520	0.00828	6.1395120
2	253815	6.091560	122640	0.06132	12480	0.01872	6.1716000
3	255351	6.128424	183240	0.09162	16920	0.02538	6.2454240
4	251997	6.047928	244800	0.12240	21600	0.03240	6.2027280
5	253632	6.087168	306600	0.15330	27000	0.04050	6.2809680
6	253632	6.087168	367920	0.18396	36000	0.05400	6.3251280
7	254265	6.102360	427560	0.21378	33600	0.05040	6.3665400
8	256044	6.145056	491520	0.24576	45120	0.06768	6.4584960
9	253623	6.086952	552960	0.27648	55080	0.08262	6.4460520
10	254274	6.102576	610800	0.30540	54000	0.08100	6.4889760
11	252906	6.069744	671880	0.33594	56760	0.08514	6.4908240
12	254265	6.102360	734400	0.36720	57600	0.08640	6.5559600
13	255105	6.122520	794040	0.39702	68640	0.10296	6.6225000
14	253917	6.094008	853440	0.42672	72240	0.10836	6.6290880
15	254580	6.109920	916200	0.45810	81000	0.12150	6.6895200
16	254241	6.101784	977280	0.48864	90240	0.13536	6.7257840
17	254349	6.104376	1038360	0.51918	104040	0.15606	6.7796160
18	254139	6.099336	1105920	0.55296	105840	0.15876	6.8110560
TOTAL	4574328	109.783872	10460760	5	943680	1.41552	116.4297720

Tabela 4.16 MODEL: iteração do modelo; TRAIN_T: quantidade de tokens treinados; TRAIN_COST: custo (em dólares) do treinamento; PROMPT_T: acumulado de tokens de requisição com mensagem ao sistema e verso a ser escandido; PROMPT_COST: custo (em dólares) das chamadas; COMP_T: acumulado de tokens com respostas dos versos escandido; COMP_COST: custo (em dólares) das respostas; TOTAL_COST: custo total de cada iteração; TOTAL: acumulados das iterações.

Tabela 4.17 Custo do modelo treinado de uma vez

TRAIN_T	TRAIN_COST	PROMPT_T	PROMPT_COST	COMP_T	COMP_COST	TOTAL_US
254193	6.100632	450560	41360	0.02068	0.67584	6.7971520

Tabela 4.18 TRAIN_COST: custo (em dólares) do treinamento; PROMPT_T: acumulado de tokens de requisição com mensagem ao sistema e verso a ser escandido; PROMPT_COST: custo (em dólares) das chamadas; COMP_T: acumulado de tokens com respostas dos versos escandido; COMP_COST: custo (em dólares) das respostas; TOTAL_COST: custo total de cada iteração; TOTAL: acumulados das iterações.

Vale salientar que este é o custo mínimo para a execução do experimento conduzido neste projeto. Outros modelos foram treinados e testados até chegarmos nesta versão *streamlined*. Chamadas à API também foram feitas em números maiores, inclusive para os experimentos de *prompt engineering*. O valor, no fim das contas, foi ainda mais alto.

Ao se projetar uma *trend line* linear (Figura 4.7) sobre o gráfico com resultado dos modelos iterativos para *match* total de strings, depuramos que, para chegar em 100%, deve-se treinar um total de 13977 exemplos (valor próximo à quantidade de versos escandidos coletados pela pesquisa). Claro, essa extrapolação nunca iria se concretizar por completo. Há nuances na própria escansão do Aoidos que impedem de haver sempre uma equivalência um para um com os versos gerados pela LLM. Haverá também sempre alguma idiosincrasia da língua que não foi captada pelo conjunto de dados. Há, por fim, a própria natureza ocasionalmente errática dos LLMs.

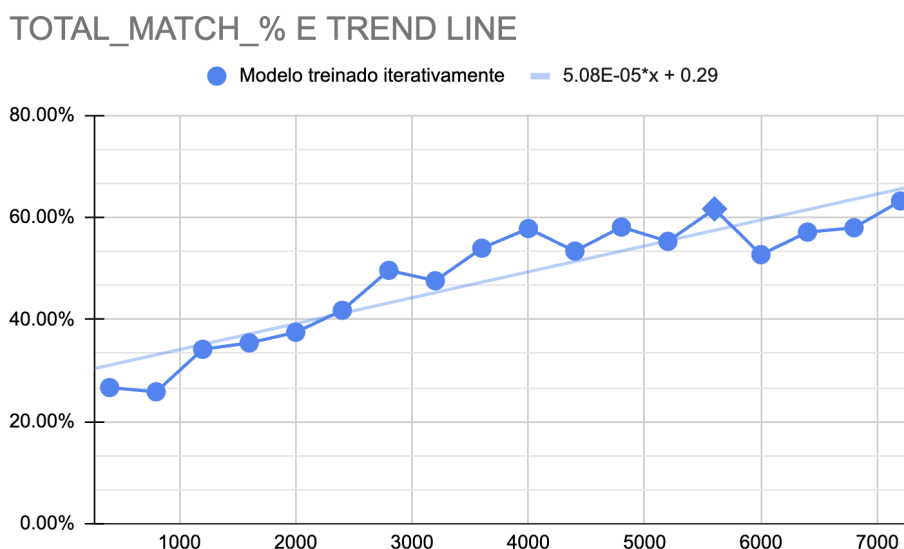


Figura 4.7 Uma tendência linear foi projetada a partir dos dados.

Ademais, só se chegaria a esse número de treinamento na 35ª iteração. Com uma média de US\$6,48 por iteração, o projeto se depararia com mais US\$110,16 de gastos. Como o custo de um experimento assim é proibitivo para o escopo do projeto (onde há naturalmente uma escassez de recursos), tivemos que impor um teto de exemplos a serem testados.

Conclusão

5.1 Observações finais

O experimento conduzido no presente projeto traz algumas revelações sobre a possibilidade de se realizar escansões de versos poéticos com o atual estado da arte das LLMs do OpenAI.

Primeiramente, fica evidente que o GPT-4 tem maior capacidade de entender o processo de escansão do que o GPT-3.5 numa situação de *prompt engineering*. Isto provavelmente se deve ao fato do GPT-4 ser um modelo mais refinado, treinado num conjunto de dados maior, com melhor capacidade de "raciocinar" o processo, principalmente quando se fornece *few-shots* de exemplos que explicam o proceder e a formatação.

Entretanto, ao tentarmos fazer a escansão via *zero-shot chain of thought*, embora haja uma aparente melhora em relação ao *zero-shot prompting*, não se chega a um resultado aceitável. Uma elucubração provável a respeito disso é que o ato de escandir um verso se trata de um problema mais fonético do que ortográfico. Disse Olavo Bilac em seu Tratado de versificação, "O ouvido (aconselha o mestre), é o melhor guia" (BILAC; PASSOS, 1905). A tese de Adiel Mittemann sobre a construção do sistema Aoidos também joga a luz nessa questão. Para atingir seu resultado, o sistema precisa fazer conversões grafema-fonema.

O que o *few-shot prompting* também demonstra é que, ao fornecer alguns poucos exemplos para a máquina, ela já é capaz de dar uma resposta mais aproximada do desejável. Ao se empregar *fine-tuning*, em que o conceito básico é o fornecimento de um alto número de exemplos, isso se torna ainda mais claro. A máquina passa a se condicionar para uma separação silábica que se aproxima do intuito esperado.

As métricas que escolhemos para avaliar a abordagem *fine-tuning* são arbitrárias, e não foram esgotadas. Haveria, possivelmente, outras maneiras de entender o *output*, como, e.g., uma extensa consulta manual, ou talvez algoritmos que tentem encontrar todas as sílabas que dão *match* entre a escansão do Aoidos e a gerada (ao invés do implementado, em que faz comparação sílaba a sílaba, linearmente). Talvez assim, a performance poderia ganhar uma migalha a mais de performance.

A título de exemplo, filtramos os resultados do modelo 14 para apenas aqueles que pontuaram 0 em *match silábico* (em tese, as piores escansões). Foram 18 no total. Em seguida, fizemos uma escansão manual muito generosa. Quanto à metrificação, foram aceitas diéreses e dialefas incomuns (em hiatos ou ditongos que dificilmente se separam; ou em momentos em que facilmente ocorreriam sinalefas); sinéreses e sinalefas forçadas (que juntam muitas vogais de uma vez); e eclipses (supressão do som nasal, provocando sinalefas), mesmo que incomuns. Do ponto de vista do esquema rítmico, observamos apenas dois aspectos: se as acentuações com "*" batem com tônicas das palavras; e se não há acentuações conjugadas. Os versos deste

conjunto foram:

1. E o Apóstolo da excelsa Caridade,
2. D’um Imortal a égide protetora
3. Justo nos feitos, no dizer prestante.
4. “Te hei com engenho e arte conduzido:
5. Giuda, Infangato: o nome os ilustrava.
6. E eu: — “Foi declarado na Escritura
7. Que um espelho é também, fará patente.” —
8. entre a gaze dos finos cortinados
9. Maia e Dione em volta pouco extensa.
10. Os hinos perturbassem, que entoavam.
11. Ao ilustre
12. Que eu aproprie da tua mente à esfera
13. Lá aos jardins da Poesia
14. — “Se eu o eterno conselho explicar ousar”
15. num esplendor de luz santa, infinita
16. O amor, que encerrava, então falando
17. “Viu-a Jacó, o topo lhe atingindo,
18. que o entusiasmo desperta,

A Tabela 5.1 explica os critérios de avaliação.

Os custos do procedimento também são um nó de tensão. Para se chegar a um conjunto de dados com a formatação mais correta possível, e um modelo mais ajustado, é necessário fazer vários outros experimentos; além de aumentar o conjunto de dados. Poderíamos ter tentado mais *system messages*, ou testado diversas combinações de hiperparâmetros, tanto no treinamento, como no *output*. Quem sabe, por exemplo, ao contrário do que se infere logicamente, altas temperaturas não são mais capazes de realizar esse processo?

Ademais, de acordo com a pesquisa realizada para o estado da arte, o projeto se coloca neste cenário de forma um tanto desbravadora. Ao afinar um LLM como o modelo GPT-3.5 para realizar uma tarefa que ainda não é capaz de cumprir sozinho, abre-se caminhos para mais experimentações em torno da geração automática de textos estruturados, principalmente os literários e poéticos.

Tabela 5.1 Exemplos frustrados

GENERATED	VALID?	REASON
E o A- / * pós- / to- / lo / da e- / * xcel- / sa / Ca- / ri- / * da- # / de,	SIM	sinérese forçada
D'um I- / mor- / * ta- / l a é- / gi- / de / pro- / te- / * to- # / ra	SIM	ectlipse
* Justo / nos / * fei- / tos, / no / di- / * zer / pres- / * tan- # / te.	NÃO	duas sílabas em uma
“Te * hei / com en- / * ge- / nho e ar- / te / con- / du- / * zi- # / do:	NÃO	tônica errada
Gi- / * u- / da, In- / fan- / * ga- / to: o / * no- / me o- / s i- / lus- / * tra- # / va.	SIM	diérese
E eu: — “ / * Foi / de- / cla- / * ra- / do / na Es- / cri- / * tu- # / ra	SIM	sinérese forçada
Que um e- / * spe- / lho é / tam- / * bém, / fa- / * rá / pa- / * ten- # / te.” —	SIM	ectlipse e sinérese forçada
en- / * tre a / gaze / dos / * fi- / nos / cor- / ti- / * na- # / dos	NÃO	duas sílabas em uma
* Mai- / a e Di- / * o- / ne em / * vol- / ta / * pou- / co ex- / * ten- # / sa.	NÃO	duas sílabas em uma
Os hi- / * nos / per- / tur- / * ba- / ssem, / que en- / to- / * a- # / vam.	NÃO	duas sílabas em uma
Ao i- / * lus- # / tre	SIM	sinérese forçada
Que eu a- / pro- / * prie / da / tua / * men- / te à es- / * fe- # / ra	SIM	sinérese forçada
* Lá / aos / jar- / * dins / da / Po- / e- / * si- # / a	SIM	diérese
— “Se eu o e- / * ter- / no / con- / * se- / lho ex- / pli- / * ca- / r ou- / * so” #	SIM	sinérese forçada
num es- / plo- / * ndor / de / luz / * san- / ta, in- / fi- / * ni- # / ta	SIM	ectlipse
O a- / * mor, / que en- / ce- / * rra- / va, en- / * tão / fa- / * lan- # / do	SIM	sinérese normal
* “Vi- / u- / a / Ja- / * có, / o / * to- / po / lhe a- / tin- / * gin- # / do,	SIM	diérese
que o en- / tu- / si- / * a- / smo / des- / * per- # / ta,	SIM	sinérese forçada

Por fim, levanta-se, comparativamente, a intenção de Mitteman ao criar o sistema Aoidos: "Seu objetivo não é gerar escansões melhores do que aquelas feitas manualmente por estudiosos; seu objetivo é gerar boas escansões [...] muito mais rápido que os estudiosos." (MITTMANN; MAIA, 2017, p. 158)

Algo parecido se aplica aqui. A língua portuguesa é múltipla, em sua expressão e nas formas fonéticas que adquire. Não há resposta definitivamente certa para se realizar uma escansão automática de um verso. Há um conjunto de sugestões baseado na fonética e em convenções culturais.

5.2 Trabalhos Futuros

É importante mais uma vez salientar que este projeto serve como pontapé para um plano maior: criar a plataforma Pajeú, que serviria como uma ferramenta de treino para poetas populares poderem exercitar seu gênio.

O modelo escansor construído aqui é, acima de tudo, um atestado de que, a depender da abordagem, as LLMs são capazes de encontrar padrões métricos nos versos. Podemos deduzir a partir disso que, se ele é capaz de encontrar, há uma configuração em que é capaz de produzir. Se não produzir, o resultado deste modelo pode ao menos servir como um "controle de qualidade", ou "filtro", para versos gerados por um outro modelo, que ficaria responsável pela parte "criativa".

Outras abordagens – utilizando LLMs ou não – podem ser usadas para engrossar esse caldo. Se *prompt engineering* não for suficiente, uma RAG talvez surgiria para resgatar informações específicas sobre a cultura, a geografia e a biologia do interior nordestino, temas recorrentes em poesias populares da região. Seria, sem isso, a máquina capaz de reproduzir os versos de Severino Milanês da Silva, pernambucano da cidade de Bezerros?

A aurora descortina,
O lampadário aparece;
Depois que o dia amanhece,
Canta o galo-de-campina;
O canário, também trina
Nas folhas dos vegetais;
Nas matas, os sabiás,
Cantando, na verde flora,
Saúdam o surgir da aurora,
Com suas notas musicais.
(WILSON, 1986)

Já o *fine-tuning* pode ser novamente aplicado para a estruturação de estrifes (décima, sextilha etc.). Uma abordagem mista (*fine-tuning*, RAG e *prompt engineering*) pode ser feita para aumentar o repertório de rimas da ferramenta, já que envolve reconhecer um padrão fonético (a rima em si), resgatar um arsenal de palavras que se aplicam ali, tudo isso de acordo com um comando fornecido pelo usuário via *prompt*. Outras ideias que não envolvem necessariamente o LLM para esses pontos de controle também são válidas.

Ferramentas que permitem uma conversa em tempo real com máquinas estão em estado de aperfeiçoamento, e muitas já se encontram disponíveis ao público, inclusive em bibliotecas *open source*¹. Elas seriam úteis para manter o caráter oral das declamações de glosas e cantos de repente nas interações, se fossem capazes de captar os detalhes regionais da oralidade.

Numa opinião pessoal do autor deste projeto, versos sintéticos nunca substituirão de fato a arte dos poetas populares. A era moderna da literatura anunciou quebras da metrificacão com o verso livre, porém, não foi capaz de frear a produçãõ de versos estruturados, cultivados quase como artesanato, de sobrevivência consuetudinária, por um segmento da populaçãõ.

A poesia é uma consequência da experiência humana, ela vincula a mente do poeta à do leitor/ouvinte. Não está só na recepção, mas ocupa todo o espaço entre a vivência do poeta, a criação, a comunicação, a recepção pelo apreciador e o legado. Fazer arte é um atestado de estar no mundo enquanto ser humano. A máquina pode espelhar o processo, mimetizar resultados, até "enganar" o apreciador; contudo, esse vínculo não se estabelece, por sua própria natureza. Embora possa comover e comunicar, é algo de outra ordem, não necessariamente inferior, somente diferente. A arte também não tem a função exclusiva de editoração, performance ou exibição; ela pode, no ato de sua feitura, ser social, terapêutica, ou lúdica. Elimina-se o elemento humano da criação, vai-se junto essa possibilidade.

Outra forma de se enxergar (um tanto generalista) é que a máquina, no fim das contas, não é o motor da sua geração. Não haveria uma diferença conceitual entre usá-la, ou se dispor de um dicionário de rimas, um compêndio de aforismos, um saco de palavras dadaístas, ou qualquer outro artifício. O importante seria que há um elemento humano que "aperta o botão", e o bônus ou ônus do resultado recairia sobre ele e/ou sobre o conjunto de forças que criou a máquina. Todavia, antes de entrar num abismo conceitual, é pertinente refrear o debate ético em torno

¹Vídeos demonstrando *low latency speech*: <https://www.youtube.com/watch?v=vgY5gNEOAZ0> , <https://www.youtube.com/watch?v=0k8wUfU7n4Q>

das IAs no contexto desse trabalho, pelo seu escopo, reduzido, e focado apenas na atividade de escansão.

Não sei ao certo se essas duas visões são antitéticas ou complementares. Sei que se o elemento humano for eliminado – em qualquer uma das pontas –, a arte vai junta no saco.

Por último – permitam-me, por favor –, gostaria de deixar abaixo uma décima de minha autoria – de versos heptassílabos e com acentuação pelo menos na terceira e sétima sílabas (3-7); muito parecido com uma redondilha maior.

1. Vaga lenta, a poesia
2. Quando pode, aperta o passo
3. Sobe, faceira, o Parnaso
4. Passa por Alexandria
5. Faz um sopro e assobia
6. Toma formas variadas:
7. De elegias a toadas,
8. Do soneto ao haiku.
9. Mas se chega ao Pajeú
10. Se detém, extasiada

Se o leitor a declamou em voz alta, percebe que é possível manipular as tônicas para atingir o padrão rítmico desejado, como convém na oralidade. Pode-se abafar um pouco a tônica de "aperta" para o verso correr parecido com os demais; e em "faceira", puxa-se a tônica para a sílaba "fa", para manter a acentuação na 3ª. Tal artimanha na 3ª linha, entretanto, não cola com o modelo escansor da 14ª iteração (ignoremos por ora errinhos na 4ª e na 8ª linha). Ao analisar verso a verso, ele coloca a tônica em seu lugar:

1. * Va- / ga / * len- / ta, a / po- / e- / * si- # / a
2. * Quan- / do / * po- / de, a- / * per- / ta o / * pa- # / sso
3. * So- / be, / fa- / * cei- / ra, o / Par- / * na- # / so
4. * Pa- / ssa / por A- / le- / xan- / * dri- # / a
5. * Fa- / z um / * so- / pro e a- / sso- / * bi- # / a
6. * To- / ma / * for- / mas / va- / ri- / * a- # / das:
7. De e- / le- / * gi- / as / a / to- / * a- # / das,

8. Do / so- / * ne- / to ao hai- / * ku. #
9. Mas / se / * che- / ga ao / Pa- / je- / * ú #
10. Se / de- / * têm, / ex- / ta- / si- / * a- # / da

Porém, a máquina não suspeita que, confrontado com esse deslize formal, o poeta tem sempre na manga a própria poesia, e a liberdade que ela lhe dá. Ele pode justificar que a quebra da métrica é intencional: faceira, garbosa e leve, a poesia se demora tranquila na sua subida ao Parnaso. Ela não tem pressa, e atrasa uma tônica.

Referências Bibliográficas

ALLARD, J.; JARVIS, C. *A Survey of Techniques for Maximizing LLM Performance*. 2023. <<https://www.youtube.com/watch?v=ahnGLM-RC1Y>>. Acesso em: 01 de março de 2024.

BILAC, O.; PASSOS, G. *Tratado de versificação*. Rio de Janeiro, Rio de Janeiro: Livraria Francisco Alves, 1905.

CALDAS, P. *Ano passado eu morri, mas esse ano eu não morro: verso absurdo une paraibanos a novas gerações*. 2021. <<https://g1.globo.com/pb/paraiba/noticia/2021/01/31/ano-passado-eu-morri-mas-esse-ano-eu-nao-morro-verso-absurdo-une-paraibanos-a-novas-geracoes.ghtml>>. Acesso em: 19 de fevereiro de 2024.

CANDIDO, A. *O estudo analítico do poema*. São Paulo, São Paulo: Editora Humanitas, 2006.

CHAKRABARTY, T.; PADMAKUMAR, V.; HE, H. *Help me write a poem: Instruction Tuning as a Vehicle for Collaborative Poetry Writing*. 2022.

CUNHA, C. F. da; CINTRA, L. F. L. *Nova gramática do português contemporâneo*. Rio de Janeiro, Rio de Janeiro: Lexikon Editora Digital, 2017.

DAVIS, N. et al. An enactive model of creativity for computational collaboration and co-creation. In: _____. *Creativity in the Digital Age*. London: Springer London, 2015. p. 109–133. ISBN 978-1-4471-6681-8. Disponível em: <https://doi.org/10.1007/978-1-4471-6681-8_7>.

GOLDSTEIN, N. *Análise do poema*. São Paulo, São Paulo: Editora Ática S.A., 1988.

GOLDSTEIN, N. *Versos, sons e ritmos*. São Paulo, São Paulo: Editora afiliada, 2005.

HERBOLD, S. et al. A large-scale comparison of human-written versus chatgpt-generated essays. *Nature*, agosto 2023.

HOLANDA, L. Ariano suassuna, cavaleiro andante da cultura brasileira. In: (ORG.), M. A. L. (Ed.). *Ode a Ariano Suassuna: celebrações dos 80 anos do autor na Universidade Federal de Pernambuco*. Recife: Editora Universitária da UFPE, 2007. p. 11–31.

HOOVER, J. *Make any large language model a better poet*. 2023. <<https://replicate.com/blog/turn-your-llm-into-a-poet>>. Acesso em: 19 de fevereiro de 2024.

KOJIMA, T. et al. *Large Language Models are Zero-Shot Reasoners*. 2023.

- LEE, T. B.; TROTT, S. *A jargon-free explanation of how AI large language models work*. 2023. <<https://developers.googleblog.com/2023/08/how-its-made-lupe-fiasco-text-fx.html>>. Acesso em: 15 de janeiro de 2024.
- LINHARES, F.; BATISTA, O. *Antologia ilustrada dos cantadores*. Fortaleza, Ceará: Imprensa universitária da Universidade Federal do Ceará, 1976.
- MITTMANN, A. *Escansão automática de versos em português*. Tese (Doutorado) — Universidade Federal de Santa Catarina, Centro Tecnológico, 2016.
- MITTMANN, A.; MAIA, S. R. Análise comparativa entre escansões manual e automática dos versos de Gregório de Matos. n. 10, julho 2017.
- ORMAZABAL, A. et al. *PoeLM: A Meter- and Rhyme-Controllable Language Model for Unsupervised Poetry Generation*. 2022.
- PIRES, R. et al. Sabiá: Portuguese large language models. In: _____. *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2023. p. 226–240. ISBN 9783031453922. Disponível em: <http://dx.doi.org/10.1007/978-3-031-45392-2_15>.
- RENZE, M.; GUVEN, E. *The Effect of Sampling Temperature on Problem Solving in Large Language Models*. 2024.
- THOMPSON, W. What we know about llms (primer). <https://willthompson.name>, July 2023. Disponível em: <<https://willthompson.name/what-we-know-about-llms-primer>>.
- WADE, A. *How it's Made: Textfx is a suite of ai tools made in collaboration with lupe fiasco*. 2023. <<https://developers.googleblog.com/2023/08/how-its-made-lupe-fiasco-text-fx.html>>. Acesso em: 15 de dezembro de 2023.
- WILSON, L. *Roteiro de velhos cantadores e poetas populares do Sertão*. Recife, Pernambuco: Centro de Estudos de História Municipal, 1986.