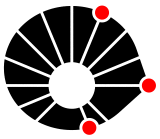

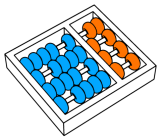

Ablação de Curadoria de Dados na Continuação de Pré-Treinamento de LLM em Português

P. Gadêlha *L. Bonifacio* *H. Pedrini* *R. Lotufo*

Relatório Técnico - IC-PFG-26-06
Projeto Final de Graduação
2026 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Ablação de Curadoria de Dados na Continuação de Pré-Treinamento de LLM em Português

Pedro Gadêlha* Luiz Bonifacio† Hélio Pedrini‡ Roberto Lotufo§

16 de julho de 2026

Sumário

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP e NeuralMind.AI.
†NeuralMind.AI.
‡Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.
§Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP. e NeuralMind.AI.

Resumo

A qualidade de um *Large Language Model* (LLM) depende fortemente dos dados da fase de pré-treinamento, e a curadoria desses dados é uma das alavancas mais acessíveis para melhorá-la, especialmente para idiomas com menos recursos que o inglês, como o português. Este trabalho estuda, de forma controlada, como diferentes métodos de *filtragem* e *deduplicação* aplicados ao corpus de continuação de pré-treinamento (*continued pre-training*, CPT) de um modelo `Qwen2.5-0.5B-Instruct` afetam, em português, a modelagem de linguagem e a capacidade de seguir instruções do modelo.

Realizamos sete experimentos de CPT (**R1**–**R7**), com o objetivo de isolar o efeito da receita de curadoria sobre cada uma dessas duas capacidades do modelo resultante. Como elementos comuns a todos eles, mantivemos os mesmos pesos iniciais e os mesmos hiperparâmetros de treinamento, sempre com uma única época, e partimos sempre do mesmo corpus inicial de cerca de 5 bilhões de *tokens*, amostrado proporcionalmente de 19 fontes do Hugging Face e de um *snapshot* de 2025 do Common Crawl, de modo que a curadoria fosse a única variável entre as execuções. O que distingue cada experimento é, portanto, apenas a receita aplicada a esse corpus, combinando métodos de deduplicação (exata e *MinHash*, em escopo global e por fonte) e de filtragem heurística de qualidade e de repetição (em versões estrita e relaxada). Para medir o impacto de cada receita, utilizamos duas famílias de métricas, uma por eixo: a modelagem de linguagem foi

medida pela perplexidade (PPL) sobre dois conjuntos *held-out* e por dois *benchmarks* de predição de última palavra (CALAME e LAMBADA-PT), ao passo que a capacidade de instrução foi medida pelo agregado do Open Portuguese LLM Leaderboard (OpenPTLLM).

Nossos resultados indicam que, em todos os sete experimentos e como pretendido, o CPT melhora a modelagem do português, o que se expressa na perplexidade e no CALAME-PT, e degrada a capacidade de instrução. A receita de melhor modelagem de linguagem, **R4** (deduplicação *MinHash* por fonte), reduz a perplexidade *held-out* em cerca de 33% em relação ao modelo sem CPT (**REF**), ao custo de cerca de 7,5% da nota de instrução; uma penalidade positiva ocorre nas sete execuções, entre 6% e 15% do agregado do REF. Observamos também um *desacoplamento* entre os eixos: a receita de melhor modelagem de linguagem (**R4**) não é a de melhor capacidade de instrução (**R1**), e tampouco é a pior. Por fim, a filtragem estrita penaliza a perplexidade, e relaxar alguns poucos limiares recupera cerca de 43% da distância que separa a filtragem estrita da melhor execução do estudo.

Palavras-chave: *continued pre-training*; deduplicação; filtragem de qualidade; corpus em português; perplexidade; avaliação de LLM.

1 Introdução

1.1 Contexto e Motivação

Os *Large Language Models* (LLMs) tornaram-se, em poucos anos, uma das tecnologias mais influentes da história, com um ritmo de adoção sem precedentes: estima-se que o ChatGPT tenha alcançado, em janeiro de 2023, cerca de 100 milhões de usuários ativos mensais, apenas dois meses após o lançamento, tornando-se, à época, a aplicação de consumo de crescimento mais rápido já registrado [17]. Esse avanço veio acompanhado de um volume de investimento igualmente expressivo: segundo o *AI Index* de Stanford, o investimento corporativo global em IA alcançou US$ 252,3 bilhões em 2024, dos quais US$ 33,9 bilhões apenas em IA generativa [25], e o investimento privado norte-americano chegou a US$ 285,9 bilhões em 2025 [35]. A tendência também alcança o Brasil: o Plano Brasileiro de Inteligência Artificial (PBIA) 2024-2028 prevê cerca de R$ 23 bilhões em investimentos e inclui, entre suas prioridades explícitas, o desenvolvimento de modelos de linguagem em português a partir de dados nacionais [7].

O desempenho desses modelos depende de maneira decisiva da quantidade, da diversidade e da qualidade dos **dados de pré-treinamento**, e não apenas da escala de parâmetros e de computação. A quantidade divulgada nos relatórios técnicos recentes já está na casa dos trilhões de *tokens*, como os 18 trilhões do Qwen2.5 [32], a própria família do modelo que utilizamos, e os cerca de 15 trilhões do Llama 3 [13], patamar que o português está longe de alcançar em corpora públicos, de modo que, sem a mesma quantidade disponível, a qualidade dos dados torna-se ainda mais decisiva. Trabalhos recentes de curadoria de dados em larga escala mostram que decisões de filtragem e de deduplicação têm efeito de primeira ordem sobre a qualidade final [28, 29], em certos casos colocando o projeto do conjunto de dados no mesmo patamar das demais decisões de projeto do modelo [22].

Esse progresso, contudo, concentra-se majoritariamente no inglês e no mandarim, idi-

omas *high-resource*, para os quais há grandes corpora bem documentados e protocolos de avaliação maduros. O português, apesar de seu grande número de falantes, ocupa uma posição de recursos intermediários: são comparativamente escassos tanto os corpora amplos e bem documentados quanto os *benchmarks* confiáveis para modelos generativos [2, 8]. Essa lacuna de recursos motiva duas frentes complementares de estudo: a construção e a curadoria de corpora em português e a adaptação de modelos já existentes por meio de continuação de pré-treinamento (*continued pre-training*, CPT), estratégia que se mostra eficaz para melhorar o desempenho em tarefas do idioma-alvo mesmo com orçamento reduzido de *tokens* [31].

Construir corpora de grande escala do zero é, no entanto, caro. Em regime de recursos limitados e escala reduzida especialmente, uma alavanca mais acessível é a curadoria de dados já disponíveis, ou seja, a aplicação de deduplicação e de filtros heurísticos de qualidade sobre coleções existentes. É essa alavanca que investigamos neste trabalho. O projeto inspira-se diretamente nos *pipelines* de curadoria do FineWeb e do FineWeb2 [28, 29] e nos esforços da série Tucano, que produziram os corpora GigaVerbo e GigaVerbo-v2 para o português [8, 9], além de trabalhos de curadoria de dados em escala industrial para o idioma [2].

1.2 Definição do Problema

O problema que este trabalho aborda é compreender como diferentes formas de curar os dados de pré-treinamento em português afetam as capacidades de um LLM submetido a continuação de pré-treinamento (CPT) com os dados curados. Concretamente, tomamos um modelo *instruct* de pequeno porte e um corpus fixo em português e variamos apenas a receita de curadoria (deduplicação e filtragem de qualidade e repetição), medindo o efeito dessas escolhas sobre o modelo resultante do CPT.

É importante distinguir o efeito esperado do CPT em si do efeito das escolhas de curadoria, o nosso objeto de estudo. Ao continuar o pré-treinamento de um modelo *instruct* sobre um corpus monolíngue, espera-se que ele melhore a modelagem de linguagem no idioma-alvo e, ao mesmo tempo, perca parte da capacidade de seguir instruções, um *trade-off* conhecido e esperado neste trabalho. Não supomos, portanto, que o desempenho em tarefas de instrução (*downstream*) acompanhe a melhora na modelagem de linguagem; tratamos os dois como eixos potencialmente independentes, a serem medidos separadamente em nossas tarefas de avaliação.

Isso define então dois eixos de avaliação. Um primeiro mede a modelagem de linguagem em português, o que o CPT pretende aprimorar, enquanto um segundo acompanha a capacidade de instrução, o que se espera que ele degrade. Assim, o problema é caracterizar como cada escolha de curadoria afeta cada eixo. Para isolar o efeito dos dados de todos os demais fatores, adotamos um desenho controlado, no espírito de estudos como o DataComp-LM [22] e o de Longpre et al. [23]: o modelo inicial e todos os hiperparâmetros de treinamento permanecem fixos entre os experimentos.

Duas perguntas principais guiam o estudo: (i) quais escolhas de deduplicação e de filtragem resultam em maior impacto positivo sobre a modelagem de linguagem em português sob CPT? e (ii) as receitas selecionadas por esse eixo-alvo preservam também o desempenho

em tarefas *downstream*, ou os dois eixos se dissociam?

1.3 Revisão Bibliográfica

A construção de corpora para pré-treinamento de LLMs a partir de dados textuais da web consolidou-se em torno de *pipelines* de curadoria compostos por extração, identificação de idioma, filtragem heurística de qualidade e deduplicação. O FineWeb [28] é uma referência metodológica dessa linha: além de mostrar que a extração do HTML bruto dos arquivos WARC supera o uso do texto pré-extraído (WET), propõe um conjunto de filtros heurísticos escolhidos pela análise das distribuições de métricas em dados de alta e de baixa qualidade, incluindo critérios de fração de linhas terminadas em pontuação, de linhas duplicadas e de linhas curtas que reutilizamos neste trabalho. Essa família de heurísticas tem raízes no Gopher/MassiveText [33], origem dos filtros de repetição e de qualidade, e no C4, cuja documentação posterior alertou para o risco de *over-filtering*, isto é, de que heurísticas agressivas removam desproporcionalmente conteúdo legítimo [10]. Pipelines abertos como o Dolma [38] e *testbeds* controlados como o DataComp-LM [22] reforçam a importância de tratar o projeto do dataset como uma variável de primeira ordem. Também, o RefinedWeb [27] soma-se a essa evidência de que a curadoria de dados afeta fortemente o desempenho dos modelos.

A deduplicação é uma segunda abordagem bem estabelecida. O trabalho de Lee et al. [21] mostra que remover duplicatas melhora a modelagem de linguagem e reduz a memorização, motivando a comparação entre métodos exatos e aproximados (*fuzzy*). Trabalhos subsequentes exploram variações: o SlimPajama [37] documenta que uma deduplicação agressiva pode encolher fortemente o corpus mantendo o desempenho; o D4 [39] incorpora diversificação no espaço de *embeddings*, indo além da coincidência de strings; e o SoftDedup [16] propõe reponderar exemplos frequentes em vez de removê-los. A escala e o escopo da deduplicação (global vs. por fonte) são, como veremos, decisões de projeto com consequências práticas.

Um terceiro conjunto de trabalhos investiga o efeito do tipo dos dados nas capacidades dos modelos. Longpre et al. [23] conduzem um estudo empírico sobre idade, toxicidade, qualidade e composição de domínios dos dados; *Textbooks Are All You Need* [14] argumenta que dados de alta densidade informacional ("qualidade de livro didático") permitem que modelos pequenos superem modelos muito maiores em tarefas de geração de código; e há esforços recentes de filtragem a nível de token [34] e de uso de LLMs para limpar dados ruidosos [15]. Esses trabalhos sustentam que decisões de curadoria não afetam o modelo de maneira uniforme, podendo moldar capacidades específicas de forma distinta.

Para o português, a série Tucano é a referência mais próxima do nosso trabalho. O Tucano [8] constrói o corpus português GigaVerbo, discutindo a escassez de corpora bem documentados e protocolos de avaliação confiáveis para o idioma; o Tucano 2 Cool [9], que produziu o GigaVerbo-v2, é a referência direta do nosso corpus, da qual espelhamos a seleção de fontes, e adota uma estratégia de deduplicação MinHash *global* para os datasets do Hugging Face e *por snapshot* para os dados de web-crawl, distinção na qual se fundamenta nossa comparação de escopo na deduplicação MinHash. O Sabiá [31] demonstra que a continuação de pré-treinamento monolíngue melhora o desempenho em tarefas locais mesmo

com um orçamento de tokens pequeno, o que justifica a adoção de CPT como método no nosso projeto.

Diante disso, o nicho deste trabalho não está em descobrir um efeito nunca visto, mas em *caracterizar, de forma controlada, como as escolhas de curadoria se comportam em escala reduzida* (um modelo de 0,5B de parâmetros e um corpus 5B de tokens) *para o português*. Vale ressaltar como nos distanciamos do FineWeb [28], nossa principal referência metodológica: em vez de treinar diversos modelos idênticos do zero, com o mesmo número de tokens e os mesmos *benchmarks*, adaptamos um único modelo pré-treinado via CPT, com um orçamento de tokens muito menor, avaliação centrada no português e em perplexidade, e sem manter o número de tokens fixo entre os experimentos. Preservamos apenas o princípio metodológico central: manter todos os fatores fixos, exceto os dados. Para favorecer a reprodutibilidade, todo o código e os resultados do estudo estão disponíveis publicamente [11].

1.4 Organização do Relatório

O restante deste relatório está organizado como segue. A Seção 2 apresenta a justificativa do trabalho, detalhando sua relevância para o português, o custo do *over-filtering* e a motivação para uma avaliação em dois eixos. A Seção 3 enuncia os objetivos geral e específicos. A Seção 4 descreve o desenvolvimento: a construção do corpus base, os três torneios de ablação (deduplicação exata vs. MinHash, escopo global vs. por fonte e filtragem estrita vs. relaxada), a combinação dos vencedores, o empacotamento, a continuação do pré-treinamento e o protocolo de avaliação. A Seção 5 apresenta e discute os resultados dos sete experimentos, incluindo o desacoplamento entre os eixos de modelagem de linguagem e de instrução. A Seção 6 sintetiza os achados, suas implicações práticas, as limitações do estudo e possíveis trabalhos futuros.

2 Justificativa

2.1 Relevância para o Português e para o Regime de Baixo Recurso

A evidência empírica sobre curadoria de dados de pré-treinamento é majoritariamente anglófona e obtida em larga escala, com corpora de trilhões de tokens e modelos grandes treinados do zero. Pouco se sabe, de forma controlada, sobre como essas escolhas se comportam para o português e em escala reduzida, cenário mais comum para grupos com orçamento computacional limitado. Quando o orçamento de tokens e de computação é pequeno, o custo relativo de coletar, armazenar e filtrar cada token é proporcionalmente maior, e uma decisão de curadoria mal calibrada pode comprometer uma fração significativa do corpus disponível.

Estudar esses efeitos especificamente para o português contribui para o esforço, ainda incipiente, de produzir datasets de pré-treinamento de qualidade para o idioma [2, 9], e o faz com um compromisso explícito de reprodutibilidade e de custo acessível.

2.2 O Custo do *Over-Filtering*

Filtros heurísticos de qualidade são, em geral, calibrados sobre dados em inglês, e seus limiares-padrão podem penalizar de forma desproporcional o texto em português, principalmente material jurídico, extraído de PDF ou denso em números e símbolos, cuja estrutura difere daquela dos dados web em inglês usados na calibração. O risco de *over-filtering* é discutido na literatura: heurísticas agressivas podem remover conteúdo legítimo e enviesar a composição do corpus [10]. Por um lado, remover texto válido reduz o volume de dados úteis, o que é especialmente danoso em regime de baixo recurso; por outro, pode distorcer a distribuição do corpus, descartando registros e domínios inteiros.

Essa preocupação motiva uma das intervenções centrais deste trabalho: em vez de aceitar os limiares canônicos, analisamos, critério a critério, o que cada filtro remove e por quê, e propomos um relaxamento controlado dos limiares mais agressivos com base na distribuição dos documentos do corpus em relação às métricas usadas pelo critério de cada filtro.

2.3 Por Que Separar Métrica Intrínseca de Extrínseca

Os dois eixos definidos na Seção 1.2 exigem instrumentos distintos, pois avaliar apenas um deles seria enganoso: uma métrica extrínseca de instrução, isoladamente, mede sobretudo a dimensão que a continuação de pré-treinamento degrada e não capta o que ela pretende aprimorar, ao passo que uma métrica intrínseca de modelagem de linguagem, isoladamente, ignoraria o custo em instrução. Por isso, adotamos uma avaliação em dois eixos, um eixo-alvo de modelagem do português e um eixo-monitor de capacidade de instrução, o que permite ler o efeito da curadoria corretamente e, sobretudo, observar quando as duas dimensões se dissociam. Essa avaliação dupla é uma contribuição metodológica do trabalho, motivada por uma constatação empírica sobre o OpenPTLLM sob CPT que detalhamos na Seção 4.10.

2.4 Utilidade Prática

Por fim, o estudo tem valor prático para outros esforços de construção de corpora de pré-treinamento em português. Ao medir, de forma controlada, o que a deduplicação e a filtragem de qualidade efetivamente entregam, o trabalho oferece subsídios para decisões de curadoria. As recomendações produzidas, baseadas em evidência reproduzível, são úteis sobretudo em contextos de recursos limitados.

3 Objetivos

3.1 Objetivo Geral

O objetivo geral do trabalho é medir, de forma controlada, como escolhas de deduplicação e de filtragem de qualidade e repetição na curadoria de um corpus em português afetam as capacidades de um LLM *instruct* de pequeno porte submetido a continuação de pré-treinamento (CPT) com esse corpus, avaliando separadamente o efeito dessas escolhas sobre a modelagem de linguagem no idioma e sobre a capacidade de seguir instruções. Todo o

restante do procedimento — modelo inicial, hiperparâmetros de treinamento e protocolo de avaliação — é mantido fixo, de modo que os dados sejam a única variável entre os experimentos.

3.2 Objetivos Específicos

Para alcançar o objetivo geral, definem-se os seguintes objetivos específicos:

- construir um corpus reproduzível de cerca de 5 bilhões de tokens em português, por amostragem proporcional das fontes escolhidas das famílias usadas no GigaVerbo-v2;
- implementar um *pipeline* reproduzível e rastreável, da construção do corpus à avaliação, com proveniência registrada por *manifests*;
- comparar métodos de deduplicação, contrastando deduplicação exata e aproximada (MinHash) em escopo global (Torneio 1) e, em seguida, escopo global e por fonte (Torneio 2);
- comparar a filtragem de qualidade e repetição em uma versão estrita e em uma versão com *thresholds* relaxados (Torneio 3), guiando o relaxamento por uma análise critério a critério do que é removido em cada filtro;
- combinar as melhores escolhas de deduplicação e de filtragem e avaliar o efeito dessa combinação;
- caracterizar, por meio de uma *suite* de avaliação em dois eixos, a relação entre modelagem de linguagem e capacidade de instrução, em particular, a eventual dissociação entre esses dois eixos e o custo de instrução (*alignment tax*).

4 Desenvolvimento do Trabalho

4.1 Visão Geral da Abordagem

A Figura 1 ilustra a abordagem experimental, composta de cinco etapas, descritas a seguir:

1. construção do corpus inicial;
2. aplicação dos métodos de curadoria escolhidos sobre o corpus;
3. empacotamento (*packing*) dos dados para o CPT;
4. execução da continuação de pré-treinamento do LLM;
5. avaliação do modelo resultante por *benchmarks* e por perplexidade sobre conjuntos *held-out*.

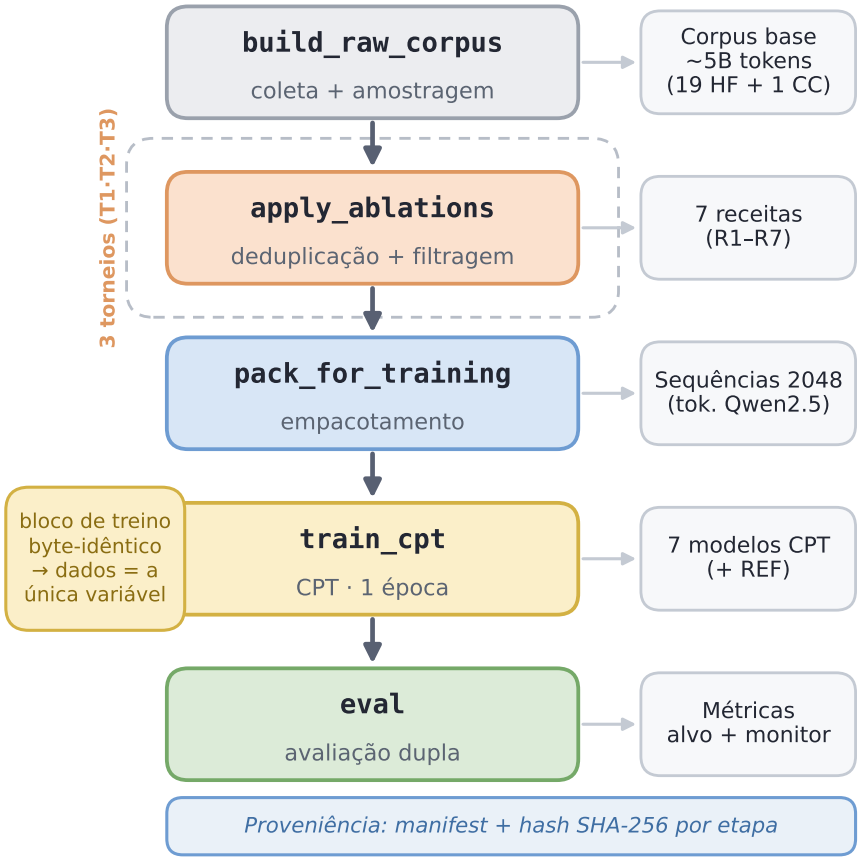

Figura 1: Visão geral da abordagem experimental. As cinco etapas processam o corpus da coleta à avaliação, cada uma produzindo o artefato à direita; a etapa de ablação instancia os três torneios (Seção 4.3). Cada etapa registra sua saída em um *manifest* com *hash* SHA-256.

Em todos os experimentos, o corpus inicial, os parâmetros de empacotamento, os hiperparâmetros da continuação de pré-treinamento e a suíte de avaliação são mantidos constantes; assim, a única variável experimental é o corpus curado. Além disso, cada execução treina sobre o respectivo corpus curado *completo*, por exatamente uma época, de modo que o número de *tokens* vistos varia entre os experimentos; a base de comparação é "uma época do corpus curado". Essa escolha é proposital e busca aproximar o estudo da prática real de treinamento: dispondo-se de dados de qualidade, deseja-se usar o máximo deles, e não descartá-los para igualar contagens. Dessa forma, o próprio *trade-off* entre quantidade e qualidade de dados passa a fazer parte do que medimos, em vez de ser eliminado por construção.

As subseções seguintes detalham cada etapa, com as decisões e descobertas feitas em sua construção e execução.

4.2 Construção do Corpus Base (∼5B tokens)

O corpus inicial é construído pela compilação de uma amostra proporcional à estimativa do tamanho total, em tokens, de 20 fontes distintas: 19 fontes do Hugging Face e um *snapshot* de 2025 do Common Crawl (CC-MAIN-2025-30), todas pertencentes às famílias adotadas na construção do GigaVerbo-v2 [9] e listadas na Tabela 1. Algumas fontes do GigaVerbo-v2 foram excluídas: HPLT1.2, cuja recuperação se mostrou lenta demais; OSCAR-2201 [1], de acesso *gated*, cuja solicitação não foi aprovada a tempo; BDTD e Baixe Livros, que exigiriam um *pipeline* de ingestão mais complexo, fora do escopo deste trabalho; e os demais *snapshots* do Common Crawl, por optarmos inicialmente por um caminho com menos dados ruidosos; decisão retomada na Seção 6.

Extraímos os dados de texto do *snapshot* do Common Crawl a partir dos arquivos WARC com o Trafilatura [4], em vez de usar os já convertidos WET, pois, como evidenciado no FineWeb [28], tal método produz dados de maior qualidade para o pré-treinamento; em seguida, filtramos o conteúdo por idioma com o FastText [20] (modelo `lid.176`/FT176), mantendo apenas os documentos classificados como português com confiança $\geq 0{,}65$, seguindo a estratégia descrita pelo GigaVerbo-v2 [9, 29].

Algumas fontes são *datasets* de instrução/*chat* (por exemplo, Dolly-15k-PT, Bactrian-X, GPT4all e UltrachatBR). Elas entram achatadas como texto puro, sem *chat template*, contribuindo como português genérico e *não* como supervisão de instrução; decisão coerente com o efeito de degradação da capacidade de instrução pretendido pelo CPT e com a composição do GigaVerbo-v2. O achatamento segue dois modos, conforme a fonte: em *datasets* de campos rotulados (como o Dolly-15k-PT, com `instruction`, `context` e `response`), os campos presentes são rotulados e concatenados, separados por uma linha em branco; em *datasets* conversacionais (como o UltrachatBR), a conversa é achatada em turnos no formato `papel: conteúdo`, também unidos por uma linha em branco.

A fim de construir um corpus menor, compatível com o nosso regime de baixo custo, mas proporcional ao que se obteria usando cada fonte em sua completude, estimamos o tamanho total em tokens de cada uma a partir das estatísticas que cada fonte publica, convertendo-as por amostragem quando necessário. Para as fontes do Hugging Face, a base de estimativa depende do que está disponível: quando há uma contagem de *tokens* publicada (FinePDFs, CulturaX), usamo-la diretamente; quando há uma contagem de *palavras* (FineWeb-2, HPLT2.0), amostramos documentos com o `tiktoken` até um teto de 100 e 50 milhões de tokens, respectivamente, para obter a razão média de tokens por palavra e a multiplicamos pela contagem publicada; e quando há apenas uma contagem de *documentos* (as demais fontes), amostramos até um teto de 500 milhões de tokens para obter a média de tokens por documento e a multiplicamos pelo número total de documentos. Para o *snapshot* do Common Crawl, partimos do índice de *WARCs* do *snapshot* (cerca de 100 000 arquivos), amostramos aleatoriamente 20 *WARCs*, aos quais aplicamos a mesma extração e filtragem de idioma descritas acima, e extrapolamos o total pela média de tokens em português por *WARC* multiplicada pelo número total de *WARCs*. Em todos os casos usamos o `tiktoken` (codificação `cl100k_base`), e não o tokenizador de treinamento (Qwen2.5), por ser um *proxy* rápido e porque precisão fina na estimativa é dispensável.

Com essas estimativas, construímos o corpus base de cerca de 5 bilhões de tokens por

amostragem proporcional de cada fonte. As cotas de proporcionalidade não têm efeito exato, pois a checagem da cota de cada fonte ocorre *após* o processamento de cada documento, de modo que a participação de cada fonte ultrapassa ligeiramente a cota estabelecida. O corpus base final contém cerca de 5,0 bilhões de tokens e 4,97 milhões de documentos (valores por fonte na Tabela 1). Inicialmente, a meta era um corpus de aproximadamente 10 bilhões de tokens; essa escala, porém, tornou o CPT demorado demais para os prazos do projeto, o que nos levou a reduzi-la para cerca de 5 bilhões.

As fontes cobrem um espectro de licenças, de permissivas a não comerciais, cujos termos de uso respeitamos; a Tabela 1 também indica a licença de cada uma.

Tabela 1: Composição do corpus base, por fonte, ordenada por número de *tokens* realizados, em milhões. A participação é a fração de *tokens* da fonte no corpus total. A família indica *HuggingFace* (HF) ou *Common Crawl* (CC). O Quati aparece agregado (soma de cinco subconjuntos). As fontes ODC-By carregam também os termos de uso do CommonCrawl. As contagens exatas de *tokens* estão no repositório do projeto [11].

Fonte	Família	Tokens (M)	Participação (%)	Documentos	Licença
HPLT2.0	HF	1 241,37	24,83	1 174 513	CC0-1.0
FineWeb-2	HF	925,85	18,52	1 054 877	ODC-By
C4	HF	687,95	13,76	786 637	ODC-By
CulturaX	HF	656,99	13,14	767 667	ODC-By
FinePDFs	HF	454,94	9,10	59 806	ODC-By
mC4-pt-cleaned	HF	369,24	7,38	485 462	ODC-By
LegalPT-dedup	HF	244,71	4,89	113 343	CC-BY-4.0
CC-MAIN-2025-30	CC	202,00	4,04	213 507	ODC-By
CrawlPT-dedup	HF	177,09	3,54	250 993	CC0-1.0
Quati	HF	15,85	0,32	46 237	CC-BY-4.0
BlogSet-br	HF	7,02	0,14	9 744	Apache-2.0
UltrachatBR	HF	5,95	0,12	3 639	MIT
Corpus-Carolina	HF	4,59	0,09	805	CC-BY-4.0
Wikipedia	HF	3,62	0,07	1 386	CC-BY-SA-3.0
GPT4all	HF	2,50	0,05	3 801	Apache-2.0
Xlsum	HF	0,31	0,01	164	CC-BY-NC-SA-4.0
Bactrian-X	HF	0,07	<0,01	268	CC-BY-NC-4.0
Dolly-15k-PT	HF	0,03	<0,01	140	CC-BY-SA-3.0
Roots-Wiki-quote	HF	0,03	<0,01	36	CC-BY-SA-3.0
Cosmos-QA-PTBR	HF	0,02	<0,01	115	CC-BY-4.0
Total		5 000,12	100,00	4 973 140	

4.3 Ablação da Curadoria de Dados

Todos os métodos de curadoria são aplicados de forma *baseada em máscara*: determinística e sem re-tokenização. Cada ablação produz uma máscara que marca os documentos não removidos, e a etapa de empacotamento consome essa máscara para gerar o conjunto de treinamento apenas com os dados que sobreviveram à receita da respectiva execução.

O corpus base, sem nenhuma curadoria aplicada, define o experimento de referência das ablações, R1 (*baseline*): o resultado da continuação de pré-treinamento sobre o corpus

base íntegro, contra o qual comparamos o efeito de cada receita para observar o efeito da curadoria.

As três subseções seguintes instanciam esse mecanismo em três torneios, organizados em duas trilhas: uma trilha de deduplicação, que encadeia o método (Torneio 1, Seção 4.4) e, em seguida, o escopo (Torneio 2, Seção 4.5); e uma trilha de filtragem, aplicada em paralelo ao mesmo corpus base, que contrasta a restritividade da filtragem de qualidade e repetição (Torneio 3, Seção 4.6). A Seção 4.7 combina os vencedores das duas trilhas. O desenho completo é esquematizado na Figura 2.

4.4 Torneio 1 — Deduplicação Exata vs. MinHash (Escopo Global)

Deduplicar dados de pré-treinamento reduz a memorização e tende a melhorar a modelagem de linguagem [21]; resta decidir *como* fazê-lo. Os primeiros dois experimentos de curadoria comparam dois métodos de deduplicação, ambos em escopo global: *hash* exato (R2) e *hash* difuso via MinHash (R3). A deduplicação exata (R2) usa *hasher* `xxh3_128` sobre o texto normalizado (NFC + colapso de espaços em branco). A deduplicação difusa (R3) usa a biblioteca `datasketch` com `threshold` 0,8, `num_perm` 128, 5-gramas de palavra e `PERM_SEED` = 1, o que a torna reprodutível byte a byte.

A deduplicação exata é a mais conservadora, removendo apenas documentos idênticos: retirou cerca de 18 milhões de tokens (0,36%), restando um corpus de cerca de 4,98 bilhões de tokens e 4,94 milhões de documentos. A abordagem por MinHash é mais agressiva, como esperado, pois, além de cobrir as duplicatas idênticas, captura as quase-duplicatas (similaridade de Jaccard $\geq 0{,}8$): removeu cerca de 244 milhões de tokens (4,87%), restando cerca de 4,76 bilhões de tokens e 4,74 milhões de documentos.

No MinHash global, a maior parte das duplicatas *near-dup* é *cross-source*, isto é, localizada em fontes distintas (90,3% dos documentos removidos), o que antecipa a remoção muito mais branda do escopo por fonte, descrita a seguir.

4.5 Torneio 2 — Escopo da Deduplicação: Global vs. Por Fonte

Para comparar escopos, o terceiro experimento de curadoria (R4) mantém o MinHash, mas com escopo *por fonte*: as duplicatas são consideradas apenas dentro de cada fonte. Preservamos os parâmetros de R3 (`datasketch`, `threshold` 0,8, `num_perm` 128, 5-gramas de palavra, `PERM_SEED` = 1). Como a maior parte das duplicatas é cross-source, o resultado é uma remoção bem mais branda: apenas cerca de 40 milhões de tokens (0,80% do total), restando cerca de 4,95 milhões de documentos (6 458 *clusters*; nenhum *cluster* cross-source, como esperado por construção, o que tomamos como verificação de sanidade), ante os 4,87% do escopo global.

Essa variação de escopo também é explorada em outros trabalhos:

- no FineWeb [28], ambos os escopos são testados, mas adota-se a deduplicação por *snapshot* do Common Crawl, pois o escopo global reduzia demais o conjunto, piorando o resultado;
- no Tucano 2 Cool [9], seguindo o FineWeb, adota-se escopo por *snapshot* para o Common Crawl e global para as fontes do Hugging Face.

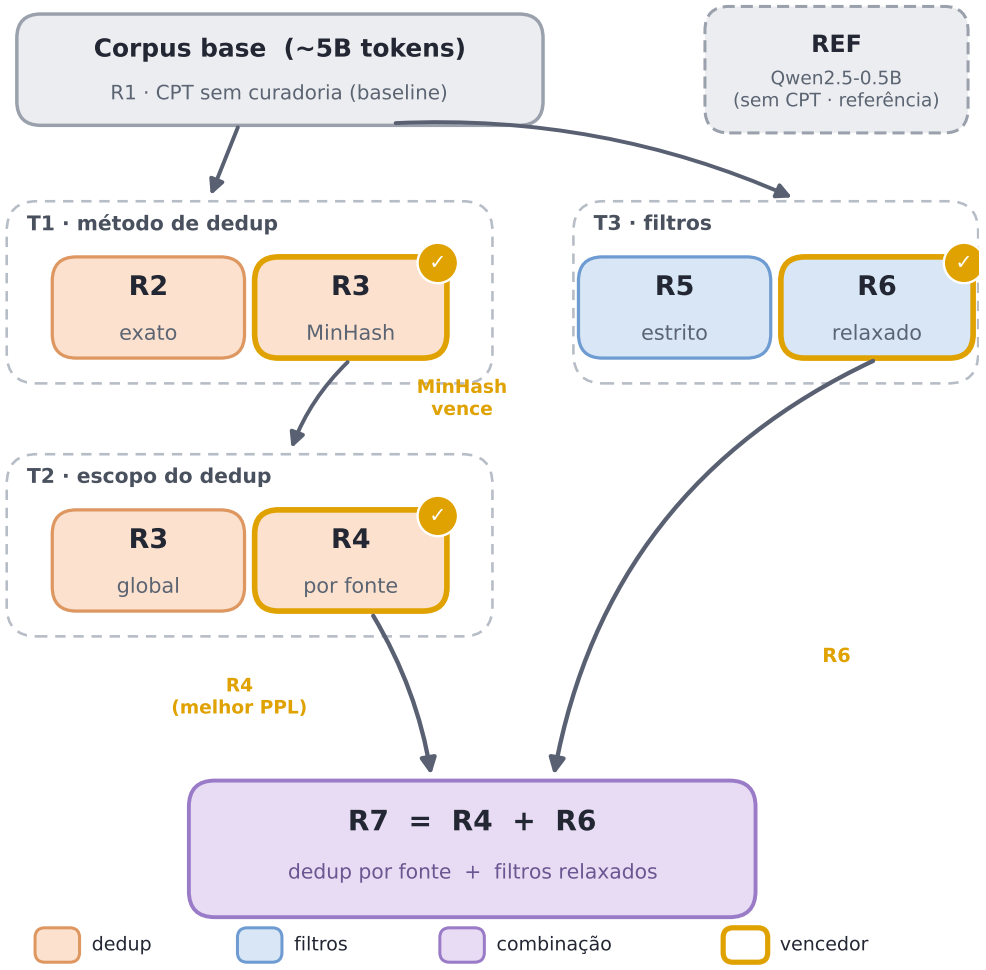

Figura 2: Desenho experimental em três torneios, organizados em duas trilhas, mais a combinação. A partir do corpus base (R1, sem curadoria), a trilha de deduplicação encadeia o Torneio 1, que compara métodos, e o Torneio 2, que compara o escopo do dedup e recebe o vencedor de T1 (MinHash); em paralelo, a trilha de filtragem instancia o Torneio 3, que compara filtros. As receitas vencedoras de T2 (R4) e T3 (R6) são combinadas em R7. REF (o modelo sem CPT) serve apenas de referência de avaliação.

4.6 Torneio 3 — Filtragem de Qualidade e Repetição: Estrita vs. Relaxada

No terceiro torneio, aplicamos ao corpus base três filtros encadeados, implementados com a biblioteca `datatrove` [30] e também empregados no Tucano 2 Cool [9]: *Gopher repetition* e *Gopher quality* [33] e *FineWeb quality* [28]. Escolhemos utilizar a lista de 207 *stopwords* do NLTK para adaptar o *Gopher quality* para o português.

O primeiro experimento deste torneio, R5, é referido como *filtros estritos*: sua curadoria removeu mais dados do que qualquer outra aplicada no projeto. Adotamos os limiares e demais parâmetros nos valores padrão das bibliotecas desses filtros, detalhados na Tabela 2; esses padrões, porém, são calibrados para o inglês. Com eles, removemos cerca de 1,80 bilhão de tokens (36,03% do total), restando um corpus de cerca de 3,20 bilhões de tokens e 3,50 milhões de documentos. O modelo treinado com esses dados obteve o pior desempenho de modelagem do português entre todos os experimentos (Seção 5); diagnosticamos, portanto, filtragem agressiva demais, confirmando empiricamente o custo de *over-filtering* antecipado na Seção 2.2.

Tabela 2: Critérios dos três filtros encadeados, com o limiar estrito (R5, padrão das bibliotecas) e o relaxado (R6). O alvo indica também a direção do critério, isto é, se o limiar é um mínimo exigido ("mín.") ou um máximo tolerado. Apenas os três critérios em destaque foram relaxados no Torneio 3; os demais mantêm o valor padrão. Os n-gramas de repetição estão agrupados por faixa.

Filtro	Critério	Alvo	Estrito	Relaxado
Gopher repetition	`dup_para_frac`	frac. parágrafos duplicados	0,30	—
	`dup_para_char_frac`	frac. caracteres em parág. dup.	0,20	—
	`dup_line_frac`	frac. linhas duplicadas	0,30	—
	`dup_line_char_frac`	frac. caracteres em linhas dup.	0,20	—
	`top_n_gram` (n=2−4)	frac. do n-grama mais comum	0,20–0,16	—
	`dup_n_grams` (n=5−10)	frac. de n-gramas duplicados	0,15–0,10	—
Gopher quality	`gopher_short_doc`	mínimo de palavras	50	—
	`gopher_long_doc`	máximo de palavras	100 000	—
	`below_avg_threshold`	compr. médio mín. de palavra	3	—
	`above_avg_threshold`	compr. médio máx. de palavra	10	—
	`too_many_hashes`	razão de "#" por palavra	0,10	—
	`too_many_ellipsis`	razão de reticências por palavra	0,10	—
	`too_many_bullets`	frac. de linhas com marcador	0,90	—
	`too_many_end_ellip.`	frac. de linhas terminadas em "..."	0,30	—
	`max_non_alpha_words_ratio`	frac. mín. de palavras com caractere alfabético	**0,80**	**0,60**
	`enough_stop_words`	mínimo de *stopwords*	2	—
FineWeb quality	`empty`	documento vazio	—	—
	`line_punct_thr`	frac. mín. de linhas com pontuação terminal	**0,12**	**0,08**
	`short_line_ratio`	frac. de linhas curtas	0,67	—
	`char_duplicates_ratio`	frac. de caracteres em linhas dup.	**0,01**	**0,05**
	`list_ratio`	razão de quebras por palavra	0,30	—

A fim de reduzir o volume removido, investigamos como relaxar os filtros de forma disciplinada. Durante a curadoria, computamos, para cada documento, as métricas de controle de cada critério de filtragem; a partir desses dados, estimamos quanto cada relaxamento isolado recuperaria. O instrumento central dessa análise é o *pool* de falhas isoladas (*sole-failure*) de um critério, ou seja, o conjunto de documentos cujo único critério violado é

aquele, que é o que se recupera ao relaxar apenas esse critério. Como os três filtros são aplicados em sequência, esse *pool* é computado no escopo de cada filtro: um documento é *sole-failure* de um critério quando aquele é o único critério violado entre os do seu próprio filtro, dentre os documentos que chegam a esse filtro. Deve-se notar que a recuperação de relaxar vários critérios em conjunto não é a soma dos *pools* isolados: por um lado, documentos que violam dois critérios ao mesmo tempo, e nenhum outro, são recuperados quando ambos são relaxados, embora não pertençam a nenhum *pool sole-failure*; por outro, um documento sole-failure de um critério de um filtro anterior, ao ser recuperado, ainda pode ser removido por um filtro posterior. Usamos o *pool* de cada critério, portanto, apenas para ordenar os critérios pela recuperação que isoladamente proporcionam, não para prever a recuperação conjunta.

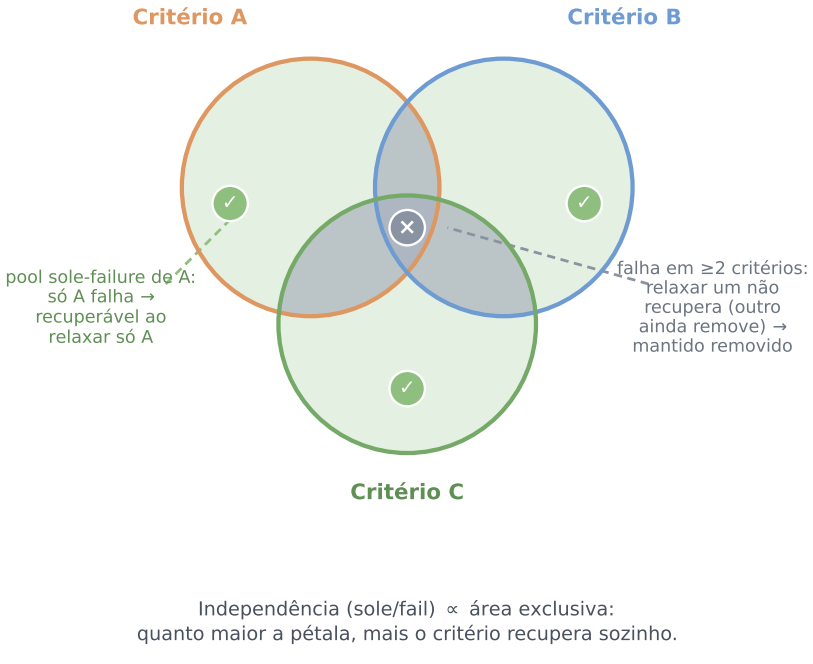

Figura 3: Conceito de *pool sole-failure* e de independência. Cada círculo é o conjunto de documentos que violam um critério de filtragem. A área exclusiva de um critério (verde) é seu *pool sole-failure*, isto é, os documentos que só violam aquele critério entre os do seu próprio filtro e que relaxá-lo isoladamente recupera nesse filtro, valor que um filtro posterior ainda pode reduzir. As interseções (cinza) reúnem documentos que violam vários critérios: relaxar um só não os recupera, pois outro ainda os remove. Quanto maior a área exclusiva, maior a independência (`sole/fail`) do critério.

Documentos que falham em múltiplos critérios ficam de fora dessa contagem por critério.

Somando os *pools* de todos os critérios, cerca de 23,03% dos tokens brutos são atribuíveis a relaxamentos isolados e ∼13% falham em múltiplos critérios simultaneamente, isto é, conteúdo de baixa qualidade genuíno, corretamente descartado. Definimos, então, o critério de decisão: relaxar apenas os critérios com *pool sole-failure* grande, isto é, os que mais recuperam quando relaxados isoladamente. Ordenando os critérios por *pool sole-failure* (Figura 3), os três critérios relaxados na receita do experimento seguinte, R6, foram:

- `max_non_alpha_words_ratio`: 0,8 → 0,6. *Pool sole-failure* de 531,8 milhões de tokens (10,64% do corpus inicial); independência ∼87%; relaxar para 0,6 recupera cerca de 516 milhões de tokens desse *pool*, valor que o encadeamento pode reduzir, dado que é critério do *Gopher quality*, anterior ao *FineWeb*. É o critério que mais retém tokens. O valor 0,8 é o padrão do Gopher para o inglês e penaliza texto denso em caracteres não-alfabéticos, como o jurídico e o extraído de PDF (números, citações, símbolos de layout). A curva de recuperação (Figura 4) concentra a massa na faixa 0,6–0,8 (documentos limítrofes): relaxar para 0,6 recupera esse conteúdo limítrofe e mantém um piso real contra ruído.

- `char_duplicates_ratio`: 0,01 → 0,05. *Pool sole-failure* de 231,2 milhões de tokens (4,62%); independência ∼80%; relaxar para 0,05 recupera cerca de 182 milhões de tokens desse *pool*. O valor 0,01 é estrito demais (descarta documentos com apenas 1% de caracteres em linhas duplicadas). Escolhemos 0,05**, e não** 0,1, deliberadamente: a análise por fonte mostrou que a faixa 0,05–0,1 é quase toda web (∼95%, sobretudo FineWeb-2 e HPLT2.0); 0,05 recupera o fundo excessivamente estrito sem abrir essa faixa de ruído.

- `line_punct_thr`: 0,12 → 0,08. *Pool sole-failure* de 184,1 milhões de tokens (3,68%); independência ∼85%; relaxar para 0,08 recupera cerca de 34 milhões de tokens desse *pool*, de modo que a maior parte dele permanece removida neste limiar. Texto extraído de PDF e com muitas linhas de layout (tabelas, títulos) tem muitas linhas sem pontuação terminal; 0,08 é conservador e fica acima do "penhasco" em ∼0,02 (Figura 4), abaixo do qual a recuperação dispara para conteúdo não-prosa genuíno.

Embora considerado, decidimos não relaxar os critérios a seguir, conforme nosso critério de decisão:

- **critérios de repetição / n-gram** (`duplicated_5..10_n_grams`, `top_2..4_gram`, `dup_line_frac`, `dup_para_frac`, `dup_*_char_frac`): *pool sole-failure* pequeno, de no máximo 38,3 milhões de tokens. Nos *n*-gramas duplicados, isso se explica pela baixíssima independência (`sole/fail` entre 1% e 6%), pois seus documentos quase sempre falham também em outros critérios; já em `dup_line_frac` (37,9%) e `top_4_gram` (33,6%) a independência é bem maior, e o que os mantém fora do relaxamento é o tamanho do *pool* em tokens. Em ambos os casos, relaxá-los recuperaria muito pouco e apenas enfraqueceria a proteção contra repetição.

- **Demais critérios do Gopher quality**: *pools sole-failure* relativamente pequenos.

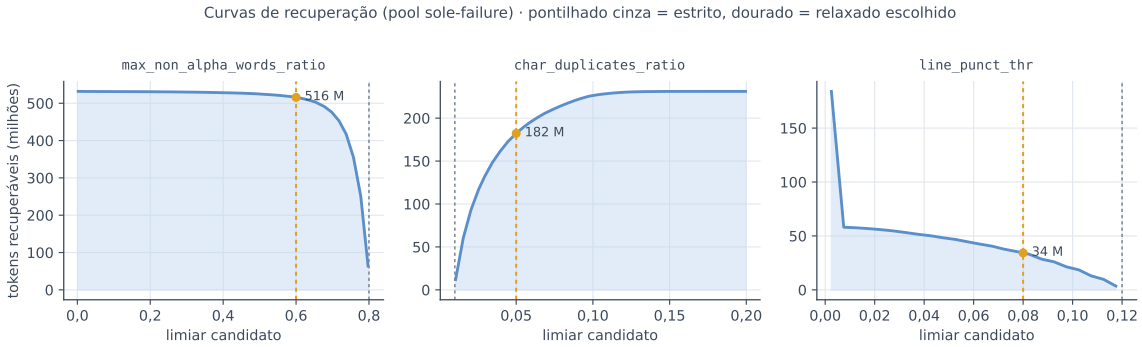

Figura 4: Curvas de recuperação dos três critérios relaxados: tokens recuperáveis do *pool sole-failure* em função do limiar candidato, a partir do valor estrito (linha pontilhada). Para o `max_non_alpha_words_ratio`, a massa recuperável concentra-se na faixa 0,6–0,8; para o `char_duplicates_ratio`, o limiar de 0,05 recupera o fundo excessivamente estrito sem abrir a faixa de ruído; para o `line_punct_thr`, a recuperação só dispara abaixo de ∼0,02, o que justifica o limiar conservador de 0,08.

Medimos a composição por fonte dos tokens na banda de relaxamento dos três *levers* (a partir dos histogramas por fonte da instrumentação, que cobrem a banda inteira, não apenas o subconjunto *sole-failure*): cerca de 77% provêm de fontes de *web* de grande escala (sobretudo HPLT2.0, C4, CulturaX e FineWeb-2, às quais se somam mC4-pt-cleaned, CrawlPT-dedup e CC-MAIN-2025-30) e cerca de 22% de fontes densas em caracteres não-alfabéticos, sobretudo texto extraído de PDF (FinePDFs) e jurídico (LegalPT). Essa composição é coerente com os critérios relaxados, que penalizam justamente excesso de repetição de *web* e alta fração de não-letras. Relaxamos apenas o grupo de qualidade (*Gopher/FineWeb quality*) e mantivemos os critérios de repetição intocados, de modo a isolar um único grupo de variáveis interpretável por ablação, preservando a disciplina de variável controlada do nosso estudo.

Com os parâmetros relaxados, R6 resulta em um corpus de cerca de 3,82 bilhões de tokens e 4,01 milhões de documentos, uma remoção de 23,52% do total; o relaxamento dos três critérios recuperou cerca de 34,7% dos tokens que R5 havia removido (Figura 5).

4.7 Combinação dos Vencedores

No último experimento, R7, combinamos os métodos de curadoria vencedores dos Torneios 2 e 3, que obtiveram melhor desempenho na modelagem do português (Seção 5): a deduplicação MinHash por fonte e a filtragem relaxada. Aplicamos a deduplicação e, sobre os documentos sobreviventes, a filtragem. A máscara de deduplicação reproduziu-se byte a byte entre R4 e R7; no total, R7 remove 23,70% do corpus.

O resultado difere pouco de R6 (que se distingue apenas pela ausência da deduplicação), o que era esperado: em R7, a deduplicação, aplicada primeiro, remove apenas 0,80% dos tokens, ao passo que a filtragem relaxada, aplicada em seguida, remove os 22,90%. O efeito dessa diferença de proporção sobre as métricas é discutido na Seção 5.

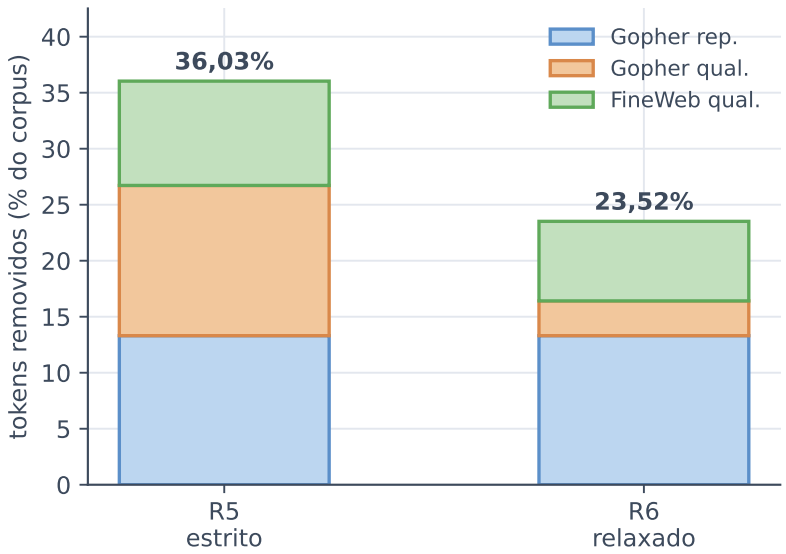

Figura 5: Remoção por etapa nos dois regimes de filtragem, estrito (R5) e relaxado (R6). Relaxar os três critérios reduz a remoção total de 36,03% (R5) para 23,52% (R6), recuperando cerca de 34,7% dos tokens que a filtragem estrita havia descartado.

4.8 Empacotamento (*Packing*)

Antes da continuação de pré-treinamento, o corpus curado de cada experimento é empacotado (*packing*) em sequências de comprimento fixo. Esta etapa consome a máscara produzida pela ablação, de modo que apenas os documentos sobreviventes à receita da respectiva execução são empacotados; e usa o mesmo tokenizador do modelo escolhido para continuação de pré-treinamento, o do Qwen2.5. Cada documento é tokenizado e seguido do separador `<|endoftext|>`; os fluxos de tokens são então concatenados e fatiados em sequências de 2 048 tokens. Os parâmetros de empacotamento são idênticos em todos os experimentos.

Duas decisões garantem determinismo e uma divisão de validação limpa. Primeiro, a ordem dos dados é definida por um embaralhamento determinístico em dois níveis, com semente fixa igual a 42, o que torna o conjunto empacotado reprodutível. Segundo, a sequência parcial final, aquela que não completa 2 048 tokens, é descartada, e uma cauda de 0,5% das sequências é reservada como divisão interna de validação, usada apenas para monitorar o treinamento e distinta da suíte de avaliação da Seção 4.10.

Como verificação de sanidade, checamos em cada execução uma identidade de conservação de tokens: o total de tokens do conjunto empacotado é igual à soma dos tokens dos documentos sobreviventes, acrescida de um separador por documento e subtraída dos tokens da sequência parcial descartada. Para o corpus base (R1), por exemplo, essa contabilidade fecha em 5 005 096 960 tokens empacotados a partir de 4 973 140 documentos. O esquema de *packing* segue a linhagem consolidada em modelos como o Megatron-LM e o GPT-NeoX [6, 36].

4.9 Continuação do Pré-Treinamento (CPT)

Cada experimento parte dos pesos pré-treinados do `Qwen2.5-0.5B-Instruct` [32], obtidos do Hugging Face[1]. É essa retomada de um modelo já treinado, e não um treinamento do zero, que caracteriza a *continuação* de pré-treinamento. A partir desse mesmo ponto de partida, para cada experimento, continuamos o pré-treinamento sobre cada conjunto empacotado usando o *framework* axolotl [3]. O bloco de hiperparâmetros é idêntico entre todos os experimentos. Treinamos por exatamente uma época sobre o corpus curado completo de cada receita, no regime definido na Seção 4.1, o que implica, por exemplo, que R6 (filtros relaxados) treina sobre mais tokens do que R5 (filtros estritos).

Os principais hiperparâmetros são um *batch* efetivo de 524 288 tokens por passo de otimização, taxa de aprendizado de 5×10^{-5} com agendamento por cosseno (*warmup* de 3% e piso de 10% do valor de pico), *weight decay* de 0,1, recorte de gradiente em 1,0 e precisão `bf16`, em uma única GPU. O *batch* efetivo decorre de três inteiros de configuração: sequências de 2 048 tokens, um *micro-batch* de 8 sequências por passo de *forward* (16 384 tokens) e 32 passos de acumulação de gradiente antes de cada atualização de pesos — ou seja, $8 \times 32 = 256$ sequências por atualização, equivalentes a 524 288 tokens ($256 \times 2\,048$). O *micro-batch* foi calibrado para o bom aproveitamento da GPU. Por depender apenas desses inteiros e do uso de uma única GPU, o *batch* efetivo é idêntico em todas as execuções, independentemente de terem ocorrido em A100 ou H100; a troca de GPU afeta apenas a vazão e o tempo de parede. Salvamos cinco *checkpoints* por execução, a cada fração de 0,2 de época, o que permite acompanhar a evolução das métricas ao longo do treinamento, e não apenas no ponto final.

O efeito pretendido do CPT, discutido na Seção 1.2, norteou o desenho do estudo e é retomado na leitura dos resultados (Seção 5.2).

Por fim, uma nota sobre a infraestrutura: no decorrer da experimentação, a máquina de treinamento passou de GPUs A100 80GB PCIe (execuções R1 e R2) para H100 NVL (R3 a R7). Essa mudança afeta apenas o *throughput* e o tempo de parede, não a qualidade dos modelos nem as métricas de avaliação, que são o que sustenta as conclusões. A Tabela 3 resume, por execução, o número de passos, os tokens vistos, a vazão em tokens por segundo, o tempo de parede e a GPU utilizada.

4.10 Protocolo de Avaliação (Suíte Dupla)

A avaliação dos modelos é feita por uma suíte dupla, idêntica em todos os experimentos, sobre os dois eixos definidos na Seção 1.2: um eixo-alvo, de modelagem do português, e um eixo-monitor, de capacidade de instrução. Submetemos à suíte os modelos resultantes das sete execuções e a referência REF, o `Qwen2.5-0.5B-Instruct` sem CPT, separando dois pontos de comparação: REF isola o efeito de *fazer* o CPT com dados em português, enquanto o *baseline* R1 isola o efeito de *curar* os dados utilizados.

Inicialmente, avaliávamos os modelos apenas pelo Open Portuguese LLM Leaderboard [12] (OpenPTLLM), um conjunto de nove tarefas voltadas principalmente à capacidade de seguir instruções. Observamos, porém, que nas primeiras execuções, quando

[1] `https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct`

Tabela 3: Custo de cada execução de CPT: GPU, número de passos de otimização até o *checkpoint* final, tokens vistos (passos × 524 288), vazão média e tempo de parede. A vazão e o tempo não são comparáveis entre execuções, pois R1 e R2 correram em A100 e R3 a R7 em H100, e o número de passos varia com o tamanho do corpus curado.

Exec.	GPU	Passos	Tokens vistos	Vazão (tok/s)	Tempo (h)
R1	A100 80GB PCIe	9 499	$4{,}98 \times 10^9$	27 980	49,4
R2	A100 80GB PCIe	9 465	$4{,}96 \times 10^9$	27 520	50,1
R3	H100 NVL	9 037	$4{,}74 \times 10^9$	39 770	33,1
R4	H100 NVL	9 423	$4{,}94 \times 10^9$	41 600	33,0
R5	H100 NVL	6 077	$3{,}19 \times 10^9$	41 270	21,4
R6	H100 NVL	7 266	$3{,}81 \times 10^9$	40 800	25,9
R7	H100 NVL	7 249	$3{,}80 \times 10^9$	41 300	25,5

dispúnhamos apenas de R1 e R2, os modelos com CPT ficavam *piores* que a REF em todas as tarefas, exceto em `assin2_sts`, por ser uma tarefa de similaridade textual semântica, a menos dependente de instrução e a mais dependente de competência linguística. Isso evidenciou que esse *benchmark*, isolado, media exatamente a dimensão que o nosso CPT degrada, e não a que ele pretende aprimorar, o que motivou a construção do eixo-alvo, de modelagem de linguagem, e o rebaixamento do OpenPTLLM à condição de monitor.

O eixo-alvo combina quatro medidas. A perplexidade é calculada sobre dois conjuntos *held-out* de 5M tokens cada: um multi-fonte, construído com a mesma composição proporcional do corpus de CPT, de modo a medir a modelagem de linguagem *in-distribution*; e um segundo extraído do Corpus Carolina, um corpus curado de qualidade relativamente aos demais, cuja perplexidade mede a modelagem sobre português de boa qualidade e serve de contraponto ao primeiro. A esses somam-se dois *benchmarks* para português de predição da última palavra, o CALAME-PT [24] e o LAMBADA-PT [26][2], que avaliam a capacidade preditiva do modelo de forma complementar à perplexidade.

Os conjuntos *held-out* são descontaminados em relação ao treinamento: aplicamos sobre eles o mesmo filtro de pertencimento por MinHash usado na deduplicação, removendo documentos próximos aos de treinamento, o que evita medir memorização em vez de modelagem. Após a descontaminação, o *held-out* multi-fonte contém ∼4,87 milhões de tokens (4 765 documentos) e o de Carolina, ∼4,82 milhões de tokens (1 057 documentos).

O eixo-monitor mantém o OpenPTLLM, com suas nove tarefas avaliadas pelo *harness* do eduagarcia; o valor agregado é a média não ponderada das nove. Neste eixo, desativamos a aplicação do *chat template* (`apply_chat_template=false`), de modo que as notas não misturem a degradação do formato de conversação sob CPT; as medidas do eixo-alvo, por construção, não usam *template*. Registramos, por fim, uma limitação de sinal deste eixo na escala de 0,5 bilhão de parâmetros: apenas quatro das nove tarefas carregam sinal apreciável; `hatebr_offensive` colapsa na classe majoritária e as tarefas de provas (`bluex`, `enem`, `oab`) situam-se perto do acaso. A discussão dos valores obtidos em cada eixo é feita na Seção 5.

[2]Empregamos a tradução automática para o português do *split* de teste do LAMBADA disponibilizada pelo projeto Tucano [8], o conjunto `TucanoBR/lambada-pt` do Hugging Face.

5 Resultados

5.1 Visão Geral

A Tabela 4 reúne, para cada uma das sete execuções de CPT (R1 a R7) e para o modelo de referência sem CPT (REF), o desempenho no *checkpoint* final nas cinco medidas da nossa suíte de avaliação: a perplexidade sobre os conjuntos *held-out* multi-fonte e de Carolina, a acurácia de predição de última palavra no CALAME-PT e no LAMBADA-PT, e o agregado de instrução do OpenPTLLM.

Tabela 4: Desempenho no *checkpoint* final das sete execuções de CPT (R1 a R7) e da referência sem CPT (REF). As colunas de perplexidade (*held-out* e Carolina) são do eixo-alvo, em que menor é melhor; as de CALAME-PT e LAMBADA-PT também são do eixo-alvo, em que maior é melhor; a de instrução é o agregado do OpenPTLLM, do eixo-monitor, em que maior é melhor.

Exec.	Receita	PPL *held-out*	PPL Carolina	CALAME	LAMBADA	Instrução
REF	sem CPT	13,013	23,882	0,4311	0,3637	0,4724
R1	*baseline*	8,673	17,719	0,4605	0,3573	0,4422
R2	dedup. exata, global	8,675	17,725	0,4619	0,3579	0,4336
R3	dedup. MinHash, global	8,672	17,722	0,4653	0,3627	0,4301
R4	dedup. MinHash, por fonte	8,653	17,708	0,4692	0,3608	0,4370
R5	filtros estritos	9,062	18,218	0,4644	0,3565	0,4268
R6	filtros relaxados	8,885	17,828	0,4692	0,3580	0,4106
R7	R4 + R6	8,884	17,831	0,4677	0,3625	0,4011

As subseções seguintes leem esses números em quatro passos, o efeito global do CPT frente ao REF (Seção 5.2), a ordenação das sete receitas entre si em cada eixo (Seção 5.3), o comportamento dos eixos ao longo do treinamento (Seção 5.4) e o custo de instrução, ou *alignment tax* (Seção 5.5). Já se nota, porém, que os melhores resultados de cada eixo não coincidem, pois a receita que mais favorece a modelagem de linguagem não é a que melhor preserva a capacidade de instrução, observação que a Seção 5.3 retoma e caracteriza.

5.2 Efeito Global do CPT (O *Trade-off* Pretendido)

O primeiro efeito que a Tabela 4 evidencia é o *trade-off* que antecipávamos para a continuação de pré-treinamento de um modelo *instruct* sobre um corpus monolíngue (Seção 4.9), manifestando-se na direção esperada em todas as sete execuções. No eixo-alvo, a modelagem do português melhora de forma expressiva e uniforme, pois a perplexidade cai cerca de um terço no *held-out* multi-fonte (de 13,013 no REF para a faixa de 8,653 a 9,062) e cerca de um quarto no conjunto de Carolina (de 23,882 para 17,708 a 18,218), enquanto a acurácia no CALAME sobe de 0,4311 para a faixa de 0,4605 a 0,4692. A Figura 6 resume esse efeito, comparando cada execução ao REF. No eixo-monitor, em contrapartida, a capacidade de instrução se degrada, dado que o agregado do OpenPTLLM de todas as execuções fica abaixo do REF (0,4724), na faixa de 0,4011 a 0,4422. Ou seja, o ganho em português vem acompanhado de uma perda de instrução, exatamente como o desenho previa, de modo

que tratamos essa degradação como um efeito esperado do CPT, e não como uma falha das receitas de curadoria.

Uma exceção a essa leitura aparece no LAMBADA-PT, a única medida do eixo-alvo que não acompanha a melhora: o REF (0,3637) permanece o maior valor, e todas as execuções ficam ligeiramente abaixo (na faixa de 0,3565 a 0,3627), ao contrário do CALAME, que sobe de maneira consistente. Vale notar que os dois *benchmarks* diferem na origem, pois o CALAME-PT é construído nativamente em português, enquanto o LAMBADA-PT que utilizamos é uma versão traduzida do LAMBADA original em inglês, e diferem também na habilidade que exigem, já que o LAMBADA depende de prever a última palavra a partir de um contexto mais longo. Não investigamos qual desses fatores explica a diferença, até porque a variação no LAMBADA-PT entre o REF e as execuções é pequena e pode estar dentro do ruído da métrica nesta escala, de modo que o ganho de modelagem de linguagem se expressa com mais clareza na perplexidade e no CALAME.

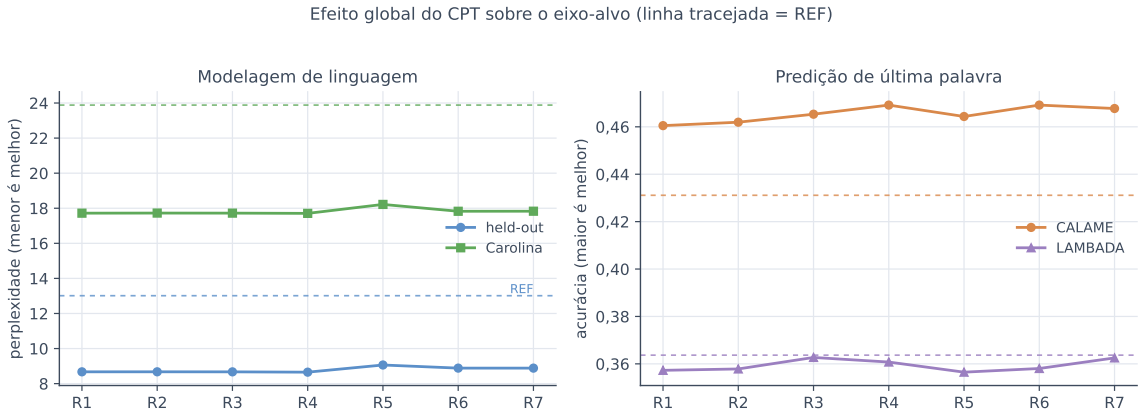

Figura 6: Efeito global do CPT sobre o eixo-alvo, por execução (R1 a R7), comparado ao REF (linha tracejada). À esquerda, a perplexidade sobre o *held-out* multi-fonte e sobre o conjunto de Carolina, em que menor é melhor; à direita, a acurácia de predição de última palavra no CALAME-PT e no LAMBADA-PT, em que maior é melhor. Todas as execuções reduzem fortemente a perplexidade e elevam o CALAME em relação ao REF, ao passo que o LAMBADA-PT permanece próximo ao valor do REF.

5.3 O Desacoplamento entre os Eixos entre Execuções

Estabelecido o efeito global do CPT, a pergunta que de fato guia o estudo é como as sete receitas de curadoria se ordenam entre si em cada eixo, e é aqui que aparece o achado central: as ordenações não coincidem.

A receita de melhor modelagem de linguagem, R4 (deduplicação MinHash por fonte, com perplexidade *held-out* de 8,653), não é a de melhor desempenho de instrução, posição que cabe a R1 (o *baseline* sem curadoria, com agregado de 0,4422), enquanto R4 fica em segundo lugar no eixo-monitor (0,4370) e R1 não é a de menor perplexidade (8,673, muito próxima da de R4). Note que não se trata então de uma simples anticorrelação, em que a

melhor de um eixo seria a pior do outro, pois a pior de instrução é R7 (0,4011), e não R4, e a pior de perplexidade é R5 (9,062), e não R1.

O que observamos é um desacoplamento, isto é, a ordem de um eixo não informa a ordem do outro, quadro que a Figura 7 torna visível ao dispor cada execução no plano da perplexidade *held-out* contra o agregado de instrução.

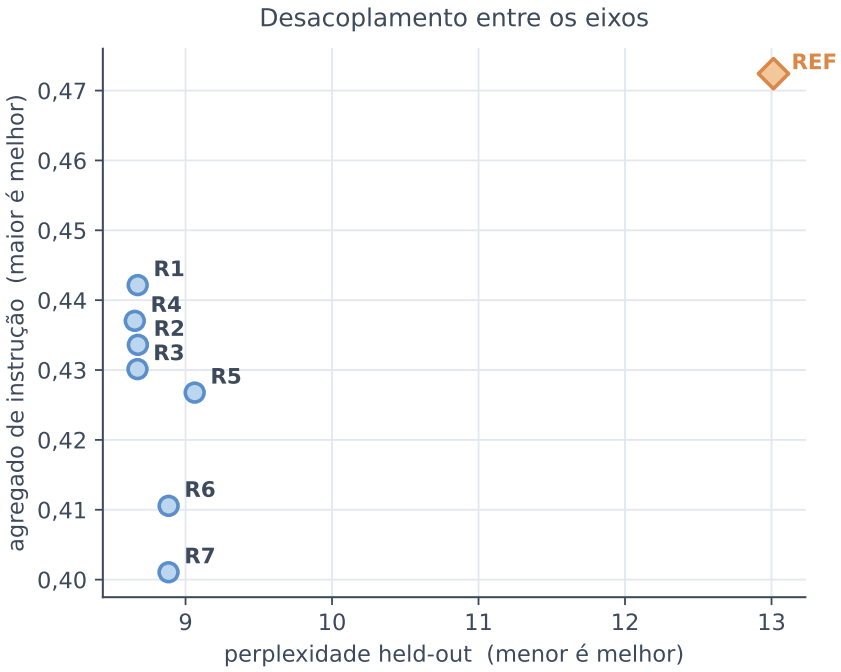

Figura 7: Desacoplamento entre os dois eixos, uma marca por execução, no plano da perplexidade sobre o *held-out* multi-fonte (eixo horizontal, menor é melhor) contra o agregado de instrução do OpenPTLLM (eixo vertical, maior é melhor). O REF é indicado como referência. A execução de menor perplexidade (R4) não é a de maior desempenho de instrução (R1), e a ordenação de um eixo não corresponde à do outro, o que caracteriza o desacoplamento.

5.4 O Desacoplamento entre os Eixos ao Longo do Treinamento

O desacoplamento entre os eixos não se manifesta apenas entre as execuções, mas também dentro de cada uma, ao longo do CPT. Como salvamos cinco *checkpoints* por execução (a cada fração de 0,2 de época), podemos acompanhar a evolução de cada eixo, e os dois se comportam de maneira distinta: a perplexidade sobre o *held-out* decresce de forma monotônica até o *checkpoint* final, ao passo que o agregado de instrução, na maioria das execuções, atinge o seu melhor valor antes do fim e depois recua. Em cinco das sete execuções (R3, R4, R5, R6 e R7), o pico de instrução ocorre em um *checkpoint* intermediário, e não no final, sendo que em R6 e R7, as duas execuções com filtragem relaxada, esse pico se dá já no primeiro *checkpoint* salvo, seguido de queda acentuada no *checkpoint* seguinte e de

recuperação apenas parcial na segunda metade do treino, que não devolve o agregado ao valor do pico (em R6, 0,4297 no passo 1453, 0,4031 no passo 2906 e 0,4106 ao final; em R7, 0,4106 no passo 1450, 0,3894 no passo 2900, 0,3864 no passo 4350 e 0,4011 ao final).

Convém notar que esse comportamento não se reduz a uma anticorrelação simples ao longo do treinamento, em que a instrução pioraria monotonicamente à medida que a modelagem de linguagem melhora. Se fosse esse o caso, o pico de instrução estaria sempre no primeiro *checkpoint*, o que ocorre apenas em R6 e R7; nas outras três execuções com pico anterior ao final (R3, R4 e R5), o pico é intermediário, isto é, a instrução primeiro sobe e só depois recai, um formato não monotônico incompatível com essa leitura. O que se observa, portanto, é a mesma dissociação da Seção 5.3, agora no eixo temporal do treinamento, pois o instante de melhor modelagem de linguagem (o *checkpoint* final) não coincide com o de melhor capacidade de instrução (um ponto anterior), de modo que treinar até o fim da época, o que otimiza o eixo-alvo, não é o que preserva melhor ou mais degrada o eixo-monitor.

Note que reportamos o *checkpoint* final em toda a nossa análise por uma questão de consistência, já que ele corresponde a uma época completa sobre o corpus curado de cada receita, mas registramos que a seleção do ponto de parada é, ela própria, uma decisão que expõe o *tradeoff*. A Figura 8 acompanha os dois eixos ao longo do treino em três execuções, R2, R4 e R6, escolhidas por exibirem o pico de instrução em pontos distintos, no *checkpoint* final, em um *checkpoint* intermediário e no primeiro *checkpoint* salvo, respectivamente.

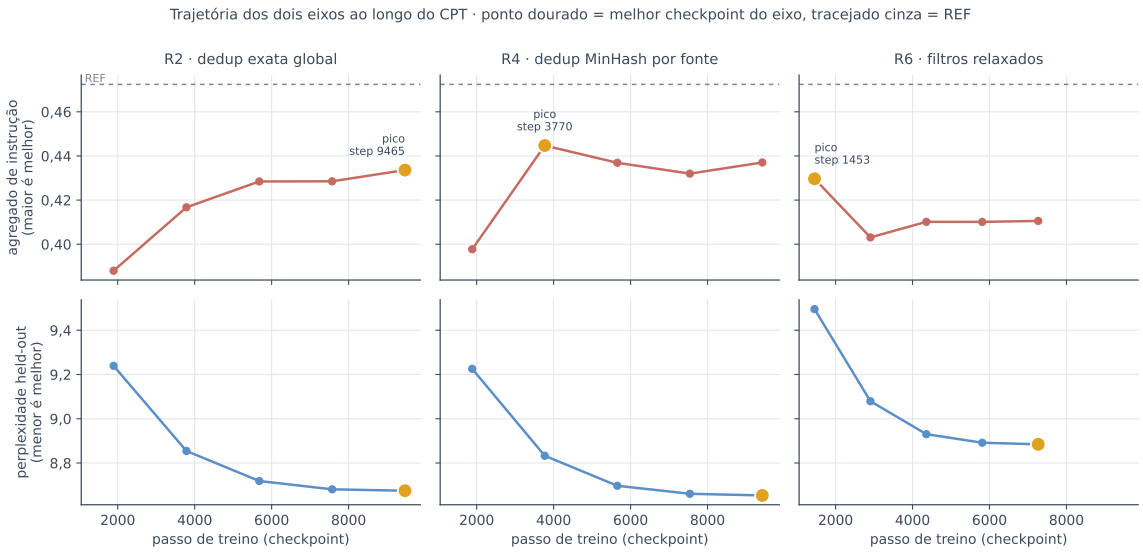

Figura 8: Trajetória dos dois eixos ao longo do CPT em três execuções, uma por coluna, com o agregado de instrução do OpenPTLLM na linha superior (maior é melhor, tracejado cinza no REF) e a perplexidade sobre o *held-out* multi-fonte na inferior (menor é melhor), e o ponto dourado marcando o melhor *checkpoint* de cada eixo. Note que a perplexidade decresce monotonicamente até o fim nas três execuções, com a mesma forma, ao passo que o pico de instrução ocorre em pontos distintos, o que caracteriza o desacoplamento.

5.5 O Custo de Instrução (*Alignment Tax*)

Estabelecido na Seção 5.2 que o CPT degrada a capacidade de instrução em todas as execuções, quantificamos aqui esse custo, o *alignment tax*[3], medido como a diferença entre o agregado do OpenPTLLM do REF (0,4724) e o de cada execução. A penalidade média é de 0,0465, cerca de 10% do agregado do REF, e varia entre 0,0303, em R1 (o *baseline* sem curadoria), e 0,0714, em R7.

Note, porém, que a magnitude dessa penalidade não acompanha o volume de dados removido por cada receita, em nenhuma das direções. Por um lado, R5, cuja filtragem estrita descarta a maior fração do corpus (36,03% dos tokens), tem penalidade menor (0,0457) que R6 (0,0619) e R7 (0,0714), que removem menos, de modo que remover mais não implicou punir mais na capacidade de instrução. Por outro lado, remover menos também não implica punir mais: R1, o *baseline* que não remove nada, tem a menor penalidade de todas (0,0303). Não há, portanto, uma relação monotônica entre o volume descartado e o custo de instrução, o que é mais uma face do desacoplamento entre os eixos, e não uma anticorrelação simples entre quantidade de dados e penalidade. Esse comportamento é retomado ao discutir o Torneio 3 (Seção 5.6), em que relaxar os filtros, apesar de preservar mais dados, aumenta a penalidade. Também vale registrar que medimos a penalidade no *checkpoint* final de cada execução, coerentemente com o restante da análise, em que o ponto de comparação é sempre o modelo após uma época completa sobre o corpus curado de cada receita. A Figura 9 apresenta a penalidade por execução, evidenciando que ela não acompanha a intensidade da remoção.

5.6 Os Torneios, Resultado a Resultado

Retomamos aqui cada torneio pelo seu resultado no eixo-alvo, que foi o critério de seleção dos vencedores (Seção 4.3). No Torneio 1, que compara os dois métodos de deduplicação em escopo global (Seção 4.4), a deduplicação por MinHash (R3, 8,672) supera a exata (R2, 8,675) na perplexidade sobre o *held-out* multi-fonte por uma margem estreita, e o CALAME aponta na mesma direção (0,4653 contra 0,4619), de modo que o MinHash vence e segue para o torneio seguinte. Note que a margem de 0,003 na perplexidade é pequena o bastante para possivelmente estar dentro do ruído da métrica nesta escala, de modo que a decisão se apoia menos no seu tamanho e mais na concordância das duas medidas do eixo-alvo. No Torneio 2, que compara o escopo da deduplicação MinHash (Seção 4.5), o escopo por fonte (R4, 8,653) supera o global (R3, 8,672) e alcança a menor perplexidade de todo o estudo, apesar de remover muito menos dados (0,80% dos tokens, contra 4,87% do escopo global), resultado que converge com a evidência do FineWeb e do Tucano 2 de que deduplicar entre fontes pode remover dados demais, ainda que a comparação não seja direta, pois esses trabalhos restringem o escopo apenas para os dados de web-crawl e mantêm o escopo global para as fontes do Hugging Face, ao passo que o nosso corpus é predominantemente composto por essas mesmas fontes do Hugging Face; e que nos leva a eleger o escopo por fonte.

O Torneio 3, que contrasta a filtragem estrita (R5) com a relaxada (R6) (Seção 4.6),

[3]Na literatura de alinhamento, esse termo pode designar o custo de capacidade imposto pelo alinhamento; aqui é invertido, pois designamos por ele o custo de alinhamento imposto pelo CPT.

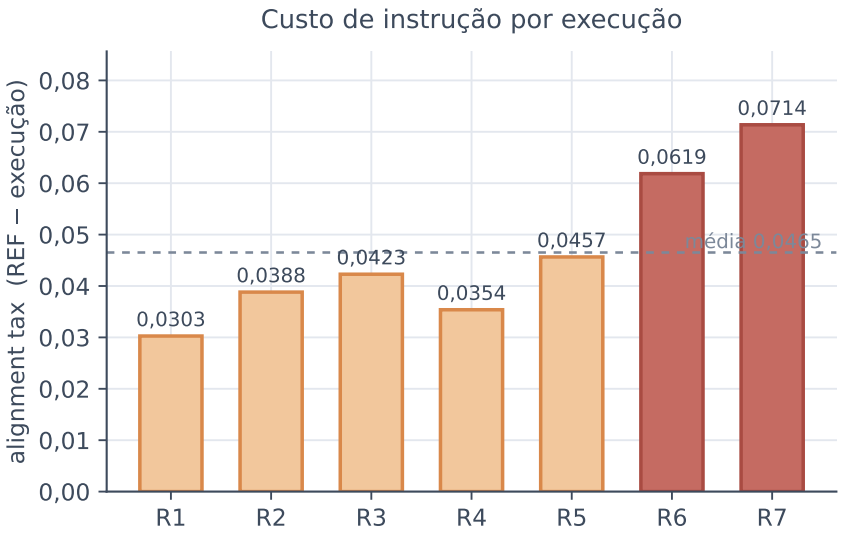

Figura 9: *Alignment tax* por execução, calculado como a diferença entre o agregado de instrução do REF (0,4724) e o de cada execução no *checkpoint* final. A penalidade é positiva em todas as sete execuções (a linha do REF é o zero de referência), com média de 0,0465. As duas maiores são as de R6 e R7, as receitas de filtragem relaxada, enquanto R5, que remove a maior fração do corpus, apresenta penalidade menor que ambas, o que mostra que o custo de instrução não acompanha o volume de dados descartado.

é o resultado central, pois é o que melhor expõe o desacoplamento sob uma intervenção controlada, já que entre as duas execuções variamos apenas a restritividade dos limiares. A filtragem estrita produz a pior perplexidade do estudo (R5, 9,062, a única acima de 9), e relaxar os limiares dos três critérios mais agressivos de forma disciplinada reduz a perplexidade para 8,885 (R6), recuperando cerca de 43% da distância que separava a filtragem estrita da melhor execução do estudo (R4, 8,653). No eixo-monitor, porém, o movimento é o oposto: o desempenho de instrução cai de 0,4268 em R5 para 0,4106 em R6. Ou seja, dentro de uma única intervenção controlada reproduz-se o desacoplamento da Seção 5.3, pois relaxar os filtros, o que melhora a modelagem de linguagem, piora a capacidade de instrução. Como a seleção é pelo eixo-alvo, R6 vence o torneio.

Por fim, a combinação dos vencedores dos Torneios 2 e 3 (Seção 4.7), a deduplicação MinHash por fonte seguida da filtragem relaxada (R7), produz um modelo cuja perplexidade (8,884) é muito próxima da de R6 (8,885), enquanto o desempenho de instrução recua de 0,4106 para 0,4011. Ou seja, acrescentar a deduplicação por fonte sobre a filtragem relaxada quase não altera a modelagem de linguagem e reduz um pouco a capacidade de instrução, efeito que atribuímos ao fato de a deduplicação por fonte remover muito pouco (cerca de 0,80% dos tokens) em relação ao que a filtragem remove (22,90%), de modo que ela pouco muda o corpus sobre o qual atua. Trata-se, portanto, de um efeito pequeno nesta escala e escopo, porém não nulo. Vale lembrar que sua magnitude tende a crescer em corpora e escopos maiores.

5.7 Análise por Tarefa e Limitações de Sinal

O agregado do OpenPTLLM, que usamos como eixo-monitor, esconde uma heterogeneidade grande entre as suas nove tarefas, e olhá-las separadamente (Tabela 5) mostra que apenas uma parte delas de fato distingue as receitas. Quatro tarefas de classificação e inferência concentram praticamente todo o poder de separação entre as execuções, a `assin2_rte`, de reconhecimento de implicação textual (que varia de 0,553 em R7 a 0,780 em R4), a `assin2_sts`, de similaridade textual semântica (de 0,480 a 0,570), a `faquad_nli`, de inferência de linguagem natural (de 0,533 a 0,610), e a `tweetsentbr`, de análise de sentimento em tuítes (de 0,338 a 0,453), ao passo que as cinco restantes pouco ou nada variam entre as execuções. Ou seja, a ordenação das execuções pelo agregado é determinada, na prática, por essas quatro tarefas.

As demais tarefas não carregam sinal apreciável nesta escala de 0,5 bilhão de parâmetros, por diferentes razões. A `hatebr_offensive` fica fixada no valor de classe majoritária (0,333) em todas as sete execuções, sem qualquer variação, e a `portuguese_hate_speech` é praticamente constante (0,412 em quase todas), de modo que ambas colapsam e não separam as receitas. As três provas, `enem_challenge`, `bluex` e `oab_exams`, situam-se na faixa de 0,27 a 0,32, próximas do acaso, que é de cerca de 0,20 no `enem_challenge` (cinco alternativas) e de cerca de 0,25 no `bluex` e no `oab_exams` (quatro alternativas), e sem ordenação estável em relação ao REF, contribuindo mais com ruído do que com sinal. Essa limitação de sinal, além do fato de as tarefas não medirem bem modelagem de linguagem, motivou, a posteriori, a construção do eixo-alvo (Seção 4.10), pois um agregado sustentado por quatro tarefas úteis, das quais só algumas se movem de forma expressiva, é um instrumento frágil para comparar receitas que diferem pouco entre si.

Mesmo assim, vale destacar duas leituras que a visão por tarefa torna possíveis. A primeira é que a `assin2_sts` é a única tarefa com sinal em que as execuções de CPT se mantêm no nível do REF (0,481) ou acima, com R6 ficando apenas 0,001 abaixo, enquanto em todas as outras com sinal o REF permanece à frente, o que é coerente com ela ser a tarefa menos dependente de seguir instruções e a mais dependente de competência linguística, exatamente a dimensão que o CPT aprimora. Ainda assim, mesmo essa tarefa sendo a mais alinhada ao nosso eixo-alvo, ela não reproduz a ordenação da perplexidade, pois, por exemplo, R4, a execução de melhor desempenho no eixo-alvo, não é a melhor na `assin2_sts` (0,565), posição que cabe a R1 (0,570). A segunda é que a maior penalidade de instrução de R6 e R7 (Seção 5.5) se concentra sobretudo na `assin2_rte`, que despenca de 0,780 em R4 para 0,619 em R6 e 0,553 em R7, o que ilustra como o custo agregado dessas receitas é, em boa parte, o efeito de poucas tarefas. Por isso, não atribuímos peso às pequenas diferenças de agregado entre execuções próximas, que podem refletir a instabilidade dessas poucas tarefas; os achados que sustentamos apoiam-se em contrastes grandes o suficiente para não dependerem dessa fragilidade, como o *trade-off* e o desacoplamento entre os eixos.

5.8 Reprodutibilidade e Proveniência

Todo o *pipeline*, da construção do corpus à avaliação, é determinístico e rastreável. Cada etapa emite um *manifest* que registra os parâmetros usados e o *hash* SHA-256 do artefato

Tabela 5: Desempenho por tarefa do OpenPTLLM, no *checkpoint* final de cada execução (R1 a R7) e no REF. As tarefas de exame (`enem`, `bluex`, `oab`) usam acurácia; a `assin2_sts` usa correlação; e as demais usam *f1-macro*. As quatro primeiras linhas de classificação/inferência concentram o poder de separação entre as execuções; `hatebr_offensive` e `portuguese_hate_speech` colapsam e as três provas ficam perto do acaso.

Tarefa	R1	R2	R3	R4	R5	R6	R7	REF
`assin2_rte`	0,778	0,770	0,699	0,780	0,760	0,619	0,553	0,826
`assin2_sts`	0,570	0,546	0,540	0,565	0,516	0,480	0,482	0,481
`faquad_nli`	0,599	0,545	0,588	0,533	0,576	0,610	0,566	0,631
`tweetsentbr`	0,453	0,449	0,451	0,438	0,338	0,370	0,362	0,546
`hatebr_offensive`	0,333	0,333	0,333	0,333	0,333	0,333	0,333	0,364
`portuguese_hate_speech`	0,412	0,412	0,412	0,412	0,412	0,412	0,411	0,468
`enem_challenge`	0,276	0,283	0,281	0,293	0,312	0,294	0,306	0,341
`bluex`	0,284	0,291	0,292	0,299	0,312	0,296	0,314	0,300
`oab_exams`	0,274	0,274	0,274	0,280	0,282	0,280	0,280	0,294

consumido e do produzido, encadeando o resultado final até o corpus que o originou. As máscaras de ablação são reproduzíveis byte a byte, o que verificamos, por exemplo, na deduplicação por fonte, cuja máscara é idêntica quando aplicada isoladamente (R4) e dentro da combinação (R7).

O bloco de treinamento é idêntico entre as sete execuções, e o empacotamento preserva a contagem de tokens, conforme a identidade de conservação verificada na Seção 4.8. Todo o código e os resultados do estudo estão disponíveis publicamente [11].

6 Conclusões

6.1 Síntese dos Achados

Este trabalho mediu como escolhas de deduplicação e de filtragem de qualidade afetam a modelagem de linguagem e a capacidade de instrução de um `Qwen2.5-0.5B-Instruct` submetido a CPT controlado em português, mantendo tudo fixo exceto os dados. O primeiro achado confirma o *trade-off* que o desenho pretendia observar: em todas as sete execuções, o CPT melhora a modelagem do português, a perplexidade *held-out* cai de 13,013 (REF) para a faixa de 8,65 a 9,06 e a acurácia no CALAME-PT sobe (o LAMBADA-PT permanece no nível do REF, ligeiramente abaixo); ao passo que degrada a capacidade de seguir instruções, com o agregado do OpenPTLLM caindo de 0,472 (REF) para valores entre 0,401 e 0,442. Ou seja, o ganho em português vem acompanhado de uma perda de instrução, exatamente na direção esperada; essa perda frente ao REF, o *alignment tax*, é, em média, de cerca de 10% do agregado do REF (aproximadamente $-0{,}0465$ em termos absolutos), e maior nas duas execuções com filtragem relaxada (R6 e R7).

O achado central, porém, está num outro nível: não no efeito do CPT frente ao REF, mas em como as sete receitas de curadoria se ordenam entre si. Comparando-as, os rankings dos dois eixos não coincidem: a receita de melhor perplexidade (R4, deduplicação MinHash por fonte, com 8,653) não é a de melhor *downstream* (R1, o *baseline* sem curadoria, com

0,442). Não se trata de uma anticorrelação simples, isto é, a de melhor perplexidade não é a de pior *downstream*, mas de um desacoplamento: a ordem de um eixo não informa a ordem do outro. Esse desacoplamento se manifesta em três níveis: entre as execuções, dentro de cada execução (o pico de *downstream* ocorre antes do *checkpoint* final em cinco das sete runs) e sob a intervenção controlada do Torneio 3, em que relaxar os filtros recupera cerca de 43% da distância de perplexidade que separa a execução estrita da melhor do estudo, mas não melhora o *downstream* (piora-o de 0,427 para 0,411). Dessa forma, escolher a receita de dados pela perplexidade não prediz qual entregará o melhor desempenho extrínseco.

Por fim, testar a combinação dos vencedores fecha o desenho experimental: acrescentar a deduplicação por fonte sobre a filtragem relaxada (de R6 para R7) altera pouco o corpus, porque remove apenas cerca de 0,80% de tokens ante os 22,90% já retirados pela filtragem. Coerentemente, quase não altera as métricas, dado que a perplexidade praticamente não se move (de 8,885 para 8,884) e o *downstream* recua de forma modesta (de 0,411 para 0,401). O efeito é, portanto, pequeno neste arranjo, e não na direção de um ganho. Isso não significa que a deduplicação seja dispensável, já que seu peso relativo é pequeno aqui apenas porque ela atua sobre o que a filtragem já reduziu, e tende a crescer em corpora e escopos maiores. Porém, avaliado com um único par de receitas, o efeito é melhor lido como um limite do nosso desenho do que como uma recomendação sobre deduplicar.

6.2 Implicações Práticas

A primeira implicação prática é que não convém escolher uma receita de curadoria olhando apenas para a perplexidade. Que o CPT degradaria a instrução já se esperava; o que o estudo acrescenta é que, entre as receitas, nem sempre o ganho de modelagem idiomática acompanha a queda de capacidade de instrução; os dois eixos se desacoplam. Como uma capacidade não pode ser inferida a partir da outra, recomendamos avaliar as duas dimensões em conjunto, a modelagem de linguagem (intrínseca) e o desempenho em tarefas (extrínseca), a fim de mirar o ponto desejado do *trade-off*, em vez de medir apenas uma delas.

A segunda implicação diz respeito à intensidade da curadoria: nos nossos experimentos, remover dados em excesso prejudicou o resultado, tanto pela deduplicação quanto pela filtragem. A recomendação prática é calibrar a agressividade de ambas com cuidado, tratando a remoção como um custo e não apenas como limpeza.

6.3 Limitações

Os resultados deste trabalho devem ser lidos à luz de algumas limitações. A mais importante é a de *escala*: os experimentos usam um único modelo de 0,5 bilhão de parâmetros e um corpus de cerca de 5 bilhões de tokens. Efeitos de curadoria de dados costumam se manifestar de forma mais nítida em escalas maiores, de modo que as diferenças modestas que observamos entre as receitas — em especial a contribuição pequena da deduplicação após a filtragem — podem crescer com mais parâmetros e mais tokens. Pelo mesmo motivo, os resultados não devem ser extrapolados diretamente para modelos ou corpora de maior porte.

A avaliação do eixo de instrução também é limitada nesta escala. Como discutido na

Seção 4.10, apenas quatro das nove tarefas do *leaderboard* carregam sinal apreciável para um modelo de 0,5 bilhão de parâmetros, enquanto as demais colapsam na classe majoritária ou situam-se perto do acaso. Isso torna o agregado *downstream* ruidoso, reduzindo a resolução com que conseguimos distinguir as receitas nesse eixo.

Além disso, o estudo cobre um único modelo, uma única escala e um único idioma, o que impede afirmar em que medida os achados se transferem para outras arquiteturas, tamanhos ou línguas. Os filtros empregados são heurísticos, e não aprendidos, e a deduplicação foi avaliada em um escopo e uma escala limitados; abordagens baseadas em modelo poderiam levar a conclusões distintas. Ainda, a comparação é feita sob o regime de "uma época do corpus curado". Como discutido na Seção 4.1, essa é uma escolha deliberada, que aproxima o estudo da prática real de aproveitar o máximo de dados de qualidade; sua contrapartida é que o número de tokens vistos varia entre as execuções, misturando os efeitos de quantidade e de qualidade dos dados, e fazendo variar também o comprimento do cronograma de taxa de aprendizado, ainda que o piso de 10% do valor de pico atenue esse efeito; tudo isso deve ser considerado na leitura dos resultados.

Por fim, o trabalho foi conduzido sob restrições de tempo e de infraestrutura: os experimentos rodaram em uma única máquina compartilhada, sob prazo. Com mais tempo ou com mais e melhores recursos computacionais, seria possível explorar diferentes composições de corpus inicial, um espaço maior de métodos de ablação, mais combinações e varreduras de parâmetros, e escalas maiores de modelo e de corpus; direções que retomamos na Seção 6.4.

6.4 Trabalhos Futuros

Este trabalho abre algumas direções de continuação. A mais imediata diz respeito à composição do corpus. Optamos por um único *snapshot* do Common Crawl e por excluir os demais, buscando um corpus inicialmente menos ruidoso; em retrospecto, essa foi uma decisão discutível, pois são justamente os dados mais ruidosos que tornam os efeitos de curadoria mais pronunciados. Incluir uma proporção maior de Common Crawl tenderia a ampliar as diferenças entre as receitas e, portanto, a dar mais resolução ao estudo, ao passo que o corpus relativamente limpo que usamos pode ajudar a explicar a magnitude modesta de vários efeitos observados. Na mesma linha, seria proveitoso experimentar diferentes composições do corpus inicial e incorporar fontes que deixamos fora do escopo, como o BDTD e o Baixe Livros, cuja ingestão exigiria um *pipeline* mais complexo.

Um segundo eixo é a exploração mais ampla do espaço de curadoria. Testamos um conjunto restrito de receitas; trabalhos futuros poderiam varrer mais combinações de parâmetros de filtragem e de deduplicação, avaliar outros métodos de deduplicação, como deduplicação semântica no espaço de *embeddings* [39] ou reponderação em vez de remoção [16]. Poderiam também aplicar curadorias distintas, com parâmetros próprios, a cada fonte do corpus: como fontes diferentes têm características de conteúdo muito distintas, uma curadoria personalizada por fonte poderia preservar melhor o conteúdo legítimo de cada uma do que limiares globais e uniformes.

Um terceiro eixo é a avaliação em maior escala. Repetir o estudo com modelos e corpora maiores permitiria verificar se as diferenças modestas observadas aqui crescem com a escala,

como sugere a literatura.

Um quarto eixo é ampliar a curadoria para além da seleção. Todas as receitas que avaliamos apenas removem documentos, mas a curadoria não precisa se restringir a descartar: trabalhos como o Cosmopedia [5], que gerou cerca de 25 bilhões de *tokens* sintéticos com um LLM aberto a partir de material semente da web, e a linha do *Textbooks Are All You Need* [14] mostram que gerar ou reescrever documentos é uma alavanca alternativa. Ou seja, em vez de descartar os cerca de 23% de *tokens* que a nossa receita relaxada remove, uma direção seria reescrevê-los com um LLM, ou gerar material sintético em português a partir deles, o que é particularmente atraente para um idioma em que a escassez de dados de alta qualidade é justamente o gargalo, ainda que o efeito dessa via sobre os dois eixos que medimos permaneça uma questão empírica em aberto.

Por fim, uma direção promissora é mitigar a degradação da capacidade de instrução em vez de apenas documentá-la. A técnica de *chat vector* [18], baseada em aritmética de vetores de tarefa [19], deriva um vetor de instrução pela diferença entre os pesos de um modelo *instruct* e os de seu modelo base correspondente e o soma aos pesos de um modelo obtido por continuação de pré-treinamento, restaurando a capacidade de seguir instruções sem treinamento adicional. A aplicação canônica soma esse vetor a um modelo *base* que passou por CPT; adaptá-la ao nosso caso exigiria, portanto, fazer o CPT a partir do `Qwen2.5-0.5B` base, e não do *instruct*, para evitar somar duas vezes a componente de instrução. Uma alternativa mais direta seria somar o vetor ao nosso próprio modelo *instruct* pós-CPT, provavelmente com um coeficiente de escala que compense essa sobreposição. Comparar as duas vias é, em si, uma questão empírica em aberto, e qualquer uma delas poderia recuperar parte do *alignment tax* que medimos, conciliando o ganho de modelagem do português com a preservação da capacidade de instrução.

Agradecimentos e Apoio

Agradecemos à NeuralMind.AI pela cessão da infraestrutura computacional utilizada neste trabalho, incluindo o acesso às máquinas com GPUs (2×H100 e 2×A100), sem as quais os experimentos de continuação de pré-treinamento não teriam sido viáveis.

Agradecemos a Luiz Bonifacio pelo acompanhamento próximo e frequente ao longo de todo o projeto, cujas orientações foram decisivas em cada etapa do trabalho. Agradecemos também ao coorientador Roberto Lotufo e ao orientador Hélio Pedrini pelas contribuições e pela orientação que tornaram este trabalho possível.

Declaração de Uso de Ferramentas de Inteligência Artificial

Ferramentas de inteligência artificial foram utilizadas como apoio ao longo deste trabalho, nas seguintes atividades: pesquisa de referências e meta-análise da literatura; codificação de *scripts*; organização e visualização de resultados; e revisão de texto e formatação em LaTeX. Os autores revisaram e validaram todo o conteúdo produzido, e os dados, experimentos e conclusões são de sua inteira responsabilidade.

Referências

[1] Abadji, J.; Ortiz Suarez, P.; Romary, L.; Sagot, B. Towards a Cleaner Document-Oriented Multilingual Crawled Corpus. In: *Proceedings of LREC 2022*. arXiv:2201.06642.

[2] Almeida, T. S.; Nogueira, R.; Pedrini, H. Building High-Quality Datasets for Portuguese LLMs: From Common Crawl Snapshots to Industrial-Grade Corpora. *Preprint arXiv*, 2025. arXiv:2509.08824.

[3] Axolotl AI. *Axolotl*: Ferramenta de Treino e *Fine-Tuning* de LLMs. Versão 0.16.1, via PyPI. Repositório: `https://github.com/axolotl-ai-cloud/axolotl`.

[4] Barbaresi, A. Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction. In: *Proceedings of ACL-IJCNLP 2021: System Demonstrations*, p. 122–131.

[5] Ben Allal, L.; Lozhkov, A.; Penedo, G.; Wolf, T.; von Werra, L. *Cosmopedia*: Conjunto de Dados Sintéticos para Pré-Treinamento. Hugging Face, fev. 2024. `https://huggingface.co/datasets/HuggingFaceTB/cosmopedia`.

[6] Black, S.; Biderman, S.; Hallahan, E. *et al.* GPT-NeoX-20B: An Open-Source Autoregressive Language Model. In: *Proceedings of BigScience Episode #5: Workshop on Challenges & Perspectives in Creating Large Language Models*, 2022, p. 95–136. arXiv:2204.06745.

[7] Brasil. Ministério da Ciência, Tecnologia e Inovação (MCTI). *Plano Brasileiro de Inteligência Artificial (PBIA) 2024–2028: IA para o Bem de Todos*. 2024. `https://www.gov.br/mcti/pt-br/acompanhe-o-mcti/transformacaodigital/plano-brasileiro-de-inteligencia-artificial`.

[8] Corrêa, N. K.; Sen, A.; Falk, S.; Fatimah, S. Tucano: Advancing Neural Text Generation for Portuguese. *Patterns* (Cell Press), 2025. arXiv:2411.07854.

[9] Corrêa, N. K.; Sen, A.; Fatimah, S.; Falk, S.; Landgraf, L.; Kastner, J.; Flek, L. Tucano 2 Cool: Better Open Source LLMs for Portuguese. *Preprint arXiv*, 2026. arXiv:2603.03543.

[10] Dodge, J.; Sap, M.; Marasović, A.; Agnew, W.; Ilharco, G.; Groeneveld, D.; Mitchell, M.; Gardner, M. Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus. In: *Proceedings of EMNLP 2021*, p. 1286–1305. arXiv:2104.08758.

[11] Gadêlha, P. *et al.* Repositório do Projeto (Código e Resultados). `https://github.com/pedrohpgadelha/ptbr-cpt-data-ablation`.

[12] Garcia, E. A. C. Open Portuguese LLM Leaderboard. Hugging Face, 2024. `https://huggingface.co/spaces/eduagarcia/open_pt_llm_leaderboard`. Avaliação sobre o *Language Model Evaluation Harness* (Gao et al.).

[13] Grattafiori, A. *et al.* The Llama 3 Herd of Models. *Preprint arXiv*, 2024. arXiv:2407.21783.

[14] Gunasekar, S. *et al.* Textbooks Are All You Need. *Preprint arXiv*, 2023. arXiv:2306.11644.

[15] He, J.; Fan, Z.; Kuang, S.; Xiaoqing, L.; Song, K.; Zhou, Y.; Qiu, X. FiNE: Filtering and Improving Noisy Data Elaborately with Large Language Models. In: *Proceedings of NAACL 2025*, p. 8686–8707.

[16] He, N.; Xiong, W.; Liu, H.; Liao, Y.; Ding, L.; Zhang, K.; Tang, G.; Han, X.; Yang, W. SoftDedup: an Efficient Data Reweighting Method for Speeding Up Language Model Pre-training. In: *Proceedings of ACL 2024*, p. 4011–4022. arXiv:2407.06654.

[17] Hu, K. ChatGPT Sets Record for Fastest-Growing User Base — Analyst Note. *Reuters*, 1 fev. 2023.

[18] Huang, S.-C. *et al.* Chat Vector: A Simple Approach to Equip LLMs with Instruction Following and Model Alignment in New Languages. In: *Proceedings of ACL 2024*. arXiv:2310.04799.

[19] Ilharco, G.; Ribeiro, M. T.; Wortsman, M.; Gururangan, S.; Schmidt, L.; Hajishirzi, H.; Farhadi, A. Editing Models with Task Arithmetic. In: *ICLR 2023*. arXiv:2212.04089.

[20] Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of Tricks for Efficient Text Classification. *Preprint arXiv*, 2016. arXiv:1607.01759.

[21] Lee, K.; Ippolito, D.; Nystrom, A.; Zhang, C.; Eck, D.; Callison-Burch, C.; Carlini, N. Deduplicating Training Data Makes Language Models Better. In: *Proceedings of ACL 2022*. arXiv:2107.06499.

[22] Li, J.; Fang, A.; Smyrnis, G. *et al.* DataComp-LM: In Search of the Next Generation of Training Sets for Language Models. In: *NeurIPS 2024*, v. 37, p. 14200–14282. arXiv:2406.11794.

[23] Longpre, S. *et al.* A Pretrainer's Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity. *Preprint arXiv*, 2023. arXiv:2305.13169.

[24] Lopes, R.; Magalhães, J.; Semedo, D. GlórIA: A Generative and Open Large Language Model for Portuguese. In: *Proceedings of PROPOR 2024*, p. 441–453. arXiv:2402.12969.

[25] Maslej, N. *et al. The 2025 AI Index Report*. Stanford Institute for Human-Centered Artificial Intelligence (HAI), 2025. `https://hai.stanford.edu/ai-index/2025-ai-index-report`.

[26] Paperno, D. *et al.* The LAMBADA Dataset: Word Prediction Requiring a Broad Discourse Context. In: *Proceedings of ACL 2016*. arXiv:1606.06031.

[27] Penedo, G.; Malartic, Q.; Hesslow, D. *et al.* The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data Only. In: *NeurIPS 2023*. arXiv:2306.01116.

[28] Penedo, G.; Kydlíček, H.; Ben Allal, L.; Lozhkov, A.; Mitchell, M.; Raffel, C.; Von Werra, L.; Wolf, T. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. In: *NeurIPS 2024*, v. 37, p. 30811–30849. arXiv:2406.17557.

[29] Penedo, G.; Kydlíček, H.; Sabolčec, V.; Messmer, B.; Foroutan, N.; Kargaran, A. H.; Raffel, C.; Jaggi, M.; Von Werra, L.; Wolf, T. FineWeb2: One Pipeline to Scale Them All — Adapting Pre-Training Data Processing to Every Language. *Preprint arXiv*, 2025. arXiv:2506.20920.

[30] Penedo, G. *et al.* (Hugging Face). *datatrove*: Biblioteca para Processamento de Dados Textuais em Larga Escala. Repositório: `https://github.com/huggingface/datatrove`.

[31] Pires, R.; Abonizio, H.; Almeida, T. S.; Nogueira, R. Sabiá: Portuguese Large Language Models. In: *BRACIS 2023*. arXiv:2304.07880.

[32] Qwen Team. Qwen2.5 Technical Report. *Preprint arXiv*, 2024. arXiv:2412.15115.

[33] Rae, J. W. *et al.* Scaling Language Models: Methods, Analysis & Insights from Training Gopher. *Preprint arXiv*, 2021. arXiv:2112.11446.

[34] Rathi, N.; Radford, A. Shaping Capabilities with Token-Level Data Filtering. *Preprint arXiv*, 2026. arXiv:2601.21571.

[35] Sajadieh, S. *et al. The 2026 AI Index Annual Report*. AI Index Steering Committee, Stanford Institute for Human-Centered Artificial Intelligence (HAI), abr. 2026. `https://hai.stanford.edu/ai-index/2026-ai-index-report`.

[36] Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; Catanzaro, B. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *Preprint arXiv*, 2019. arXiv:1909.08053.

[37] Soboleva, D.; Al-Khateeb, F.; Myers, R.; Steeves, J. R.; Hestness, J.; Dey, N. SlimPajama: A 627B-token, cleaned and deduplicated version of RedPajama. Cerebras (blog técnico), jun. 2023. `https://www.cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama`.

[38] Soldaini, L.; Kinney, R.; Bhagia, A. *et al.* Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. In: *Proceedings of ACL 2024*. arXiv:2402.00159.

[39] Tirumala, K.; Simig, D.; Aghajanyan, A.; Morcos, A. D4: Improving LLM Pretraining via Document De-Duplication and Diversification. In: *NeurIPS 2023*. arXiv:2308.12284.