What Do Prompts Reveal About Model Capabilities in Low-Resource Languages?

Oluwaseun A. Ajayi
The African Research Collective
oluwaseunajayi248@gmail.com

Odunayo Ogundepo
The African Research Collective
odunayo.ogundepo@taresco.org

Abstract

Large language models (LLMs) are highly sensitive to prompt design, yet benchmark evaluations typically rely on static, hand-crafted instructions that may underestimate true model capability. In this work, we study a reflective prompt evaluation algorithm, GEPA as an inference-time optimization strategy across multiple multilingual benchmarks spanning diverse African languages. GEPA uses a more capable model as a reflection agent to iteratively optimize prompts under strict compute and latency budgets, without updating model parameters.

Results show that reflective prompt optimization consistently improves performance across tasks, enabling smaller models to match or outperform larger models when evaluated using optimized instructions. We find that prompt evolution functions as a form of textual policy learning, improving not only task accuracy but also output structure and formatting factors that are critical for reliable model evaluation. Qualitative analysis further demonstrates that optimized prompts elicit more stable model behavior. We characterize the trade-off between optimization cost and inference latency by measuring prompt token growth, and show that modest increases in prompt length can yield substantial gains in performance. Based on these findings, we argue that benchmark evaluations should report both baseline and prompt-optimized results to more faithfully reflect model capabilities, particularly in multilingual and low-resource settings.

1 Introduction

Large language models (LLMs) have enabled a wide range of NLP tasks to be performed directly through prompting, but their performance is often sensitive to prompt phrasing and formatting (Anagnostidis and Bulian, 2024; Razavi et al., 2025). This effect becomes even more brittle in multilingual and low-resource settings, where limited training data, orthographic variation, and tokenizer issues can amplify instruction-level effects. While some prior work suggests that text generation tasks such as machine translation are relatively insensitive to prompt variation (Ojo et al., 2025), these findings arise from comparisons among prompt variants that are near-semantic rewrites of one another and differ only in surface form. For African languages, recent benchmarks such as IrokoBench and AfroBench demonstrate that even strong models often underperform or exhibit inconsistent behavior across tasks and languages (Adelani et al., 2025; Ojo et al., 2025; Adebara et al., 2025). In these settings, small changes in prompt structure, especially those that clarify translation literalness, diacritic handling, or task expectations, can result in large differences in evaluation metrics, suggesting that prompt choice may substantially influence reported performance.

Most existing approaches to improving LLM performance in low-resource languages focus on continuous pre-training and fine-tuning (Lu et al., 2024; Akera et al., 2025; Yu et al., 2026). While effective, these approaches can be costly, difficult to reproduce, or infeasible when high-quality labeled data is scarce. In contrast, prompt optimization offers an alternative to inference-time, enabling adaptation without modifying the model weights. Recent prompt optimization methods (Opsahl-Ong et al. (2024); Chen et al. (2024); Agrawal et al. (2025)) have shown promise across a variety of tasks, yet their behavior on African language benchmarks, as well as the trade-off between performance gains and added inference cost, remains underexplored.

In this work, we study GEPA (Agrawal et al., 2025), a reflective prompt optimization framework. GEPA treats prompts as policies that can be iteratively improved using task feedback and textual reflection, enabling a form of inference-time improvements without gradient updates. We do

not introduce a new optimization method. Instead, we investigate whether reflective prompt evolution should be considered a competitive alternative to fine-tuning and a meaningful evaluation baseline when assessing model capabilities in low-resource multilingual settings. We evaluate GEPA on select benchmarks spanning multiple African languages and tasks using the DSPy framework(Khattab et al., 2024) for our experiments. Operating under strict optimization budgets, we ask: to what extent can feedback-driven prompt evolution alone close performance gaps in African-language benchmarks, and what latency costs measured in prompt token growth do they incur?

Our findings show that prompt optimization can substantially improve performance, often enabling smaller models to perform on par with or better than larger models evaluated with standard prompts, with the largest gains observed in less capable models. Based on these results, we argue that benchmark evaluations for African languages and low-resource languages more broadly should report prompt-optimized results alongside baseline prompts. Without such reporting, comparisons across models risk conflating prompt design choices with underlying model capability, obscuring both progress and limitations in low-resource multilingual NLP.

2 Automatic Prompt Optimization Techniques

Automatic prompt optimization methods treat prompts as parameters that can be iteratively refined and optimized using task-level feedback, rather than relying solely on manual prompt engineering. Recent approaches such as MIPROv2 (Opsahl-Ong et al., 2024), MAPO (Agrawal et al., 2025) and GEPA (Agrawal et al., 2025) among others, share similar logic: a base prompt is evaluated on training examples, and execution traces or failures are analyzed, candidate prompt updates are proposed, and only revisions that improve performance are retained. These methods differ primarily in how prompt updates are generated and how candidates are selected, but at a high level they all implement reflection-driven, inference-time prompt improvement without updating model weights.

In this work, we adopt GEPA as a representative and lightweight instantiation of this paradigm. As summarized in Algorithm 1, GEPA iteratively refines prompts by using a reflection model to propose text-level modifications conditioned on execution traces and task feedback, retaining only candidates that yield measurable improvements. Unlike gradient-based approaches, GEPA operates entirely in discrete prompt space and naturally supports multi-objective selection, such as accuracy and latency, making it well suited for multilingual and low-resource settings where supervision is sparse and evaluation budgets are constrained. Our study therefore focuses on characterizing how far such reflective prompt refinement can improve performance on African-language benchmarks, and the associated prompt-length latency trade-offs.

Algorithm 1 Pseudocode for GEPA (Agrawal et al., 2025). GEPA iteratively selects candidate prompts under a fixed optimization constraint, samples minibatches from the training set, and applies reflective updates using execution traces. A new prompt is added to the candidate pool only if it improves performance on the sampled data.

Require: Initial prompt π_0; training set $\mathcal{D}_{\text{train}}$; validation set $\mathcal{D}_{\text{val}}$; task model M; reflection model R; batch size b; optimization budget B

Initialize candidate pool $\mathcal{P} \leftarrow \{\pi_0\}$
Evaluate π_0 on $\mathcal{D}_{\text{val}}$ and store score vector
Initialize total used budget $c \leftarrow 0$
while $c < B$ **do**
  Select candidate $\pi \in \mathcal{P}$ using Pareto-based selection
  Sample minibatch $\mathcal{B} \subset \mathcal{D}_{\text{train}}$, $|\mathcal{B}| = b$
  Run model M with prompt π on $\mathcal{B}$ and collect outputs, execution traces, and task-level feedback
  Compute average score s_{before} on $\mathcal{B}$
  Use reflection model R to generate an updated prompt π' conditioned on π, traces, and feedback
  Run model M with prompt π' on $\mathcal{B}$
  Compute average score s_{after} on $\mathcal{B}$
  $c \leftarrow c + |\mathcal{B}|$
  if $s_{\text{after}} > s_{\text{before}}$ **then**
    Add π' to candidate pool $\mathcal{P}$
    Evaluate π' on $\mathcal{D}_{\text{val}}$ and store score vector
  end if
end while
return $\arg\max_{\pi \in \mathcal{P}} \mathbb{E}_{x \sim \mathcal{D}_{\text{val}}}[\text{score}(M(\pi, x))]$

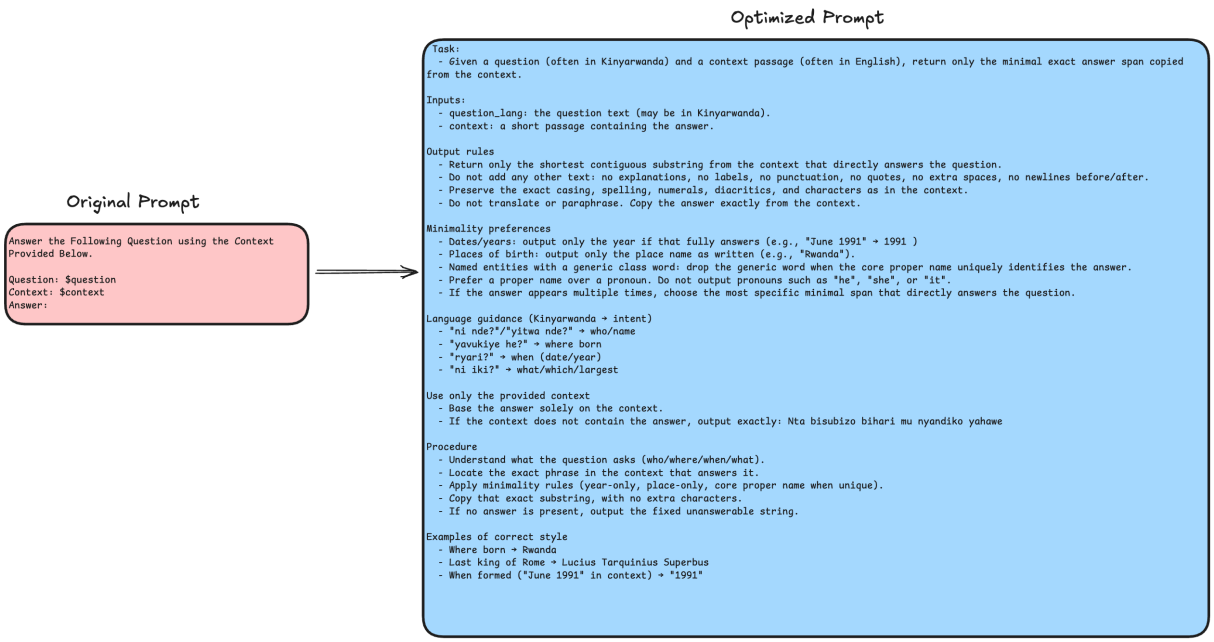

Figure 1: An example showing reflective prompt evolution for AfriQA (Kinyarwanda, `kin`). GEPA expands a minimal baseline instruction into a language and task-constrained extraction prompt for `gpt-3.5-turbo`. This refinement improves Exact Match from **11.6** to **29.9** and F1 from **18.8** to **41.0**, demonstrating how inference-time prompt optimization can substantially improve model performance. Additional optimized prompt examples across tasks are provided in Appendix B.

3 Experimental Setup

For all tasks and languages, we apply prompt optimization using GEPA under strict budgets using similar configurations across all tasks. By strict budget, we mean that prompt optimization is performed using a fixed and limited number of labeled examples and iterations, without using the full training set. Specifically, for AfriQA Yoruba, the dataset contains 247 training samples and 253 test samples. From these, we sample 50 examples from the training split to serve as the prompt optimization set and an additional 50 examples from the same split for prompt candidate selection, while reserving the full test split for final evaluation. Other tasks follow the same protocol, with task-dependent subset sizes determined by the available data, and using a validation split for candidate selection when one is provided.

Unless otherwise stated, GPT-5 serves as the reflection model, proposing prompt changes based on execution traces and task-level feedback on the training and development set. We additionally report some ablations with self-reflection, in which the same model serves as both the task executor and the reflection model, using an otherwise identical configuration. This allows us to isolate the contribution of reflective prompt evolution itself from the use of a more capable external reflector. For each task, the inputs, metrics, feedback functions and scoring logic used during prompt optimization are described in detail in Appendix A.

3.1 Tasks

Below is a list of tasks we cover:

AfriMMLU (Adelani et al., 2025) This dataset consists of multiple-choice questions from multiple domains that have been translated into different African languages. We evaluate Hausa, Igbo, Kinyarwanda, Lingala, Swahili, Twi, Wolof, and Yorùbá.

Commonsense physical reasoning (PIQA) (Chang et al., 2025). PIQA is a binary-choice goal–solution dataset where each example consists of a goal, two candidate solutions, and a correct label. In this task, we test whether automatic prompt optimization improves a model's ability to map naturalistic goals to plausible physical actions in an African-language setting. For this task, we consider only the Yoruba language.

AfriQA (Ogundepo et al., 2023) We evaluate cross-lingual question answering using the AfriQA gold-passage setting. This dataset covers Bemba, Fon, Hausa, Igbo, Kinyarwanda, Twi, Yorùbá, and Zulu. Each instance consists of a question in an

			Exact Match (EM)									F1								
Model	**Setting**	**Reflection Model**	Yor	Hau	Ibo	Zul	Fon	Bem	Kin	Twi	Avg	Yor	Hau	Ibo	Zul	Fon	Bem	Kin	Twi	Avg
gemma 3 12B	Base	–	20.2	32.7	49.6	32.3	9.3	10.7	25.5	14.6	24.4	34.6	40.1	64.9	45.2	17.4	15.1	38.2	20.3	34.5
gemma 3 12B	Optimized	gpt-5	32.4	48.7	55.0	39.7	10.9	18.5	37.4	29.4	34.0	45.7	57.6	68.8	53.5	17.4	23.2	51.5	38.4	44.6
gpt-4.1-mini	Base	–	25.3	39.0	46.5	24.0	8.6	12.0	26.4	26.5	26.0	43.4	51.7	65.1	41.4	15.6	20.9	46.0	38.9	40.4
gpt-4.1-mini	Optimized	gpt-4.1-mini	36.8	48.0	56.2	34.8	10.4	18.5	35.4	37.9	34.7	54.0	55.2	73.1	51.0	21.3	31.3	50.5	49.6	48.2
gpt-4.1-mini	Optimized	gpt-5	**41.1**	**50.5**	**59.2**	**38.5**	**16.1**	**22.7**	**40.3**	**45.3**	**39.2**	**57.8**	**59.6**	**74.4**	**54.8**	**23.9**	31.2	**58.2**	**55.9**	**52.0**
gpt-3.5-turbo	Base	–	11.5	18.3	18.8	17.9	6.0	10.4	11.6	9.7	13.0	21.8	24.4	30.7	27.8	12.6	16.7	18.8	16.2	21.1
gpt-3.5-turbo	Optimized	gpt-3.5-turbo	15.4	20.7	22.5	17.5	10.4	15.9	19.7	14.6	17.1	24.6	27.0	35.1	28.6	17.1	21.2	27.5	18.2	24.9
gpt-3.5-turbo	Optimized	gpt-5	17.8	33.0	35.7	29.9	8.3	16.6	29.9	26.5	24.7	27.6	40.1	44.7	40.7	11.4	22.3	41.0	34.4	32.8
gpt-3.5-turbo	Twi-Optimized	gpt-5	21.0	22.3	31.8	26.2	10.4	17.2	23.5	26.5	22.4	31.6	28.7	42.8	36.5	16.4	23.5	30.1	34.4	30.5
gpt-4.1-mini	Twi-Optimized	gpt-5	36.8	44.3	58.4	42.8	14.0	20.8	34.5	45.3	37.1	51.4	52.2	73.7	57.1	22.0	29.4	49.6	55.9	48.9
gpt-5	Base	–	30.8	36.3	42.5	16.6	11.4	15.9	25.8	29.0	26.1	53.6	54.2	67.2	43.6	21.5	34.6	53.1	46.4	46.8

Table 1: Exact Match (EM) and F1 results on AfriQA. 'Setting' denotes whether the base program, an optimized per-language program, or a program optimized on Twi was used. 'Reflection Model' indicates the model used during optimization.

African language, a passage in English or French, and one or more gold answers from the passage. We evaluate answers in each individual language allowing us to access end-to-end QA capabilities of the model including translation from English to French

Machine translation (AfriDocMT). To study prompt optimization for translation, we use AfriDocMT (Alabi et al., 2025), a document machine translation corpus constructed from public-domain reports. We focus on language pairs involving English and Hausa, Swahili, and Yorùbá, evaluating both *En* $\rightarrow$ *X* and *X* $\rightarrow$ *En* directions depending on the experimental setting. Rather than translating entire documents, we evaluate translation quality on shorter passages consisting of up to five consecutive sentences sampled from each document.

Reading comprehension (Belebele). Finally, we include the Belebele multiple-choice reading comprehension dataset (Bandarkar et al., 2024), selecting African language variants Yorùbá, Hausa, Igbo, Zulu, Fon, Kinyarwanda, Bemba, and Twi. Each example consists of a short passage, a question, and four answer options.

3.2 Models

We explore prompt optimization in two regimes:

- Cross-Model Reflection, where a stronger model provides feedback to improve a weaker executor, and
- Self Reflection, where a single model serves as both executor and reflection model.

As base (executor) models, we evaluate gpt-3.5-turbo, gpt-4.1-mini, and Gemma 3 12B. These represent a spectrum of capability and cost, and are typical choices for multilingual evaluation in practice. We primarily use gpt-5 as the reflection model in the cross-model setting, i.e., GEPA improves prompts for a fixed executor model using feedback from gpt-5. To investigate a more realistic low-resource scenario where access to a much stronger model is limited or unavailable, we also run GEPA in a *self-reflection* configuration: the same model (gpt-3.5-turbo, gpt-4.1-mini, or Gemma 3 12B) is used both to execute the task and to generate feedback for prompt prompt updates. This setup is conceptually related to recent work on self-distillation in reinforcement learning, which shows that models can leverage their own feedback to iteratively refine behavior without relying on an external teacher or reward model (Hübotter et al., 2026). In particular, Self-Distillation Policy Optimization (SDPO) demonstrates that conditioning a model on rich feedback enables it to identify and correct its own mistakes, effectively turning the model into a self-teacher.

In all cases, model weights are frozen; only the textual prompts are updated. We use the public `dspy.GEPA` implementation, with its `auto="light"` configuration and a fixed train and validation samples per task–language pair, to ensure comparable optimization cost across models and datasets.

4 Results and Discussion

4.1 AfriQA: Per-Language vs. Shared Optimized Programs

Table 1 reports the exact match (EM) and F1 results in AfriQA. Across all evaluated models, per-language GEPA optimization leads to large and consistent improvements over the base prompt, with especially pronounced gains for smaller and weaker models.

Model	Setting	Reflection Model	Accuracy								
			Yor	Hau	Swa	Twi	Ibo	Kin	Wol	Lin	Avg
gemma 3 12B	Base	–	23.5	31.5	40.0	16.5	29.7	31.2	12.7	21.7	25.8
gemma 3 12B	Optimized	gpt-5	44.0	49.7	62.5	33.5	52.0	52.7	34.5	46.7	46.9
gpt-4.1-mini	Base	–	54.8	62.6	67.8	45.8	51.2	64.2	35.4	56.2	54.8
gpt-4.1-mini	Optimized	gpt-4.1-mini	61.6	67.2	72.4	48.2	67.0	65.8	41.6	58.6	60.3
gpt-4.1-mini	Optimized	gpt-5	56.2	70.6	68.4	49.2	53.8	64.2	43.6	60.4	58.3
gpt-3.5-turbo	Base	–	14.0	17.6	34.0	8.8	17.0	13.2	6.6	17.2	16.1
gpt-3.5-turbo	Optimized	gpt-3.5-turbo	28.2	28.0	43.0	15.6	23.2	19.0	17.8	26.8	25.2
gpt-3.5-turbo	Optimized	gpt-5	33.6	27.6	48.0	32.0	33.4	28.8	30.4	37.2	33.9
gpt-5	Base	–	**82.8**	**83.8**	**91.2**	**71.8**	**86.8**	**82.6**	**65.2**	**84.8**	**81.1**

Table 2: Results on AfriMMLU (accuracy, %). Setting indicates whether evaluation was performed using the base (unoptimized) program or a GEPA-optimized program; for optimized runs, the Reflection Model specifies the model used to generate reflective feedback. Bold indicates best per column; underline indicates second-best per column.

Model	Setting	Reflection Model	Yor
gpt-4.1-mini	Base	–	66.0
gpt-4.1-mini	Optimized	gpt-4.1-mini	72.0
gpt-4.1-mini	Optimized	gpt-5	68.0
gpt-3.5-turbo	Base	–	46.0
gpt-3.5-turbo	Optimized	gpt-3.5-turbo	43.0
gpt-3.5-turbo	Optimized	gpt-5	51.0
gpt-5	Base	–	**92.0**

Table 3: Results on PIQA-Yoruba (accuracy, %). Setting indicates whether evaluation was performed using the base (unoptimized) program or a GEPA-optimized program; for optimized runs, the Reflection Model specifies the model used to generate reflective feedback.

Model	Setting	Reflection Model	Accuracy				
			Yor	Hau	Ibo	Zul	Swa
gpt-4.1-mini	Base	–	53.8	72.0	56.5	67.7	84.0
gpt-4.1-mini	Optimized	self	57.5	73.0	58.0	64.5	83.0
gpt-4.1-mini	Optimized	gpt-5	54.0	71.2	60.0	68.2	82.7
gpt-3.5-turbo	Base	–	1.7	1.7	0.2	2.0	9.2
3.8							
gpt-3.5-turbo	Optimized	self	11.5	27.2	11.7	23.0	39.0
gpt-3.5-turbo	Optimized	gpt-5	18.0	27.0	19.0	30.0	64.7
gpt-5	Base	–	**79.5**	**87.5**	**79.0**	**81.7**	**94.2**

Table 4: Results on Belebele (accuracy, %). We report base (unoptimized) and prompt-optimized variants; for optimized runs, the 'Reflection Model' indicates the model used to generate reflective feedback (self-reflection uses the same model). Bold indicates best per column; underline indicates second-best per column.

Per-language optimization. For **gpt-4.1-mini**, the base prompt averages 26.0 EM across languages. Applying GEPA with self-reflection increases this to 34.7 EM, while cross-reflection using GPT-5 further improves performance to 39.2 EM (Table 1). Notably, in several languages under AfriQA the self and cross reflection variants outperform the raw GPT-5 baseline.

The effects are more pronounced for **gpt-3.5-turbo**. Here, GEPA roughly doubles performance: average EM rises from 12.1 with the base prompt to 17.1 with self-reflection and 23.6 with GPT-5 reflection, while average F1 improves from 19.6 to 24.9 and 31.7, respectively. The largest gains occur in languages where the base prompt performs poorly (e.g., Fon and Bemba), highlighting the fragility of naive prompting in low-resource settings.

Gemma 3 12B sits between these two models: its base prompt is stronger than gpt-3.5-turbo but weaker than gpt-4.1-mini, and cross reflection yields substantial improvements (from 24.4 to 34.0 average EM, 34.5 to 44.6 F1), though still slightly below the best gpt-4.1-mini cross reflection configuration. This suggests that reflective prompt optimization can meaningfully lift an open-weight executor as well, not only proprietary GPT variants.

For AfriQA, although the supporting passages are provided in English, we require the model to produce its final answer in the language of the query. This design choice intentionally evaluates the model's ability to both identify the correct answer from the passage and express it in the target African language, a behavior that was inconsistent prior to optimization.

Shared Twi-optimized programs. Beyond per-language optimization, we evaluate a *shared optimization* setting in which GEPA is run only on Twi, and the resulting Twi-optimized program is reused for all other languages (rows marked *Twi-Optimized* in Table 1.

We choose Twi as the source language because it combines some of the largest relative gains from GEPA across model variants with moderate-to-high absolute EM/F1 scores: languages such as Fon or Bemba exhibit very large percentage improvements but remain low in absolute performance, whereas

			doc_health_5						doc_tech_5					
Model	**Setting**	**Reflection Model**	En→Yo	Yo→En	En→Ha	Ha→En	En→Sw	Sw→En	En→Yo	Yo→En	En→Ha	Ha→En	En→Sw	Sw→En
gemma 3 12B	Base	–	25.03	47.86	44.28	56.23	65.56	73.40	26.01	46.41	46.29	58.90	60.70	67.74
gemma 3 12B	Optimized	gpt-5	30.70	50.54	44.80	57.47	65.27	74.56	30.90	48.18	46.77	59.62	61.10	68.00
gpt-4.1-mini	Base	–	34.62	60.19	52.60	58.20	68.73	73.43	35.45	56.33	53.60	61.19	59.49	
gpt-4.1-mini	Optimized	gpt-5	**41.81**	**62.77**	52.50	60.42	69.74	**75.42**	40.65	56.76	54.23	61.36	67.11	
gpt-3.5-turbo	Base	–	17.08	34.09	31.07	36.46	61.93	59.58	18.39	35.15	34.10	40.27	52.82	
gpt-3.5-turbo	Optimized	gpt-5	24.60	37.80	34.99	**41.36**	**68.57**	72.20	24.06	39.58	37.09	44.83	62.73	

Table 5: AfriDocMT (bi-directional) results using ChrF++ (higher is better). 'Setting' indicates whether evaluation was performed using the base (unoptimized) program or a GEPA-optimized program; for optimized runs, the 'Reflection Model' specifies the model used to generate reflective feedback.

Twi improves substantially while ending up among the strongest languages overall

For **gpt-4.1-mini**, this single optimized program achieves an average EM of 31.9 across the remaining seven languages, compared to 26.0 for the base prompt and 39.2 for full per-language cross reflection optimization. For instance, EM in Yorùbá improves from 25.3 (base) to 36.8 (Twi-Optimized), and further to 41.1 with per-language optimization; in Hausa, performance increases from 39.0 to 44.3 and then to 50.5.

A similar pattern is observed for **gpt-3.5-turbo**, where the Twi-Optimized configuration attains an average EM substantially higher than the base prompt, though still below the per-language cross reflection result.

Overall, these results suggest that a *single* optimized program derived from a relatively strong language can transfer effectively to other African languages on the same task. While per-language optimization yields the best performance, shared optimization recovers a substantial fraction of the GEPA gains while requiring only a single optimization run instead of one per language. Practically, shared optimization is also far cheaper: one GEPA run on Twi replaces eight separate per-language runs, substantially reducing both optimization time and token cost.

Self-Reflection vs Cross-Reflection. Comparing rows within each model, self-reflection consistently helps but cross-reflection with GPT-5 helps more. For gpt-4.1-mini, self-reflection recovers roughly two-thirds of the EM gain that full cross-reflection achieves; for gpt-3.5-turbo, the gap between self and cross reflection is larger. Intuitively, weaker executors benefit more from a stronger teacher. At the same time, self-reflection requires no additional model and yields shorter programs (subsection 4.4), making it an appealing option when access to a strong reflection model is constrained.

4.2 Knowledge and Reading-Comprehension Benchmarks

Table 2 shows that on AfriMMLU, prompt optimization improves all base models but does not close the gap to gpt-5. For gpt-4.1-mini, self-reflection gives the best small-model average (54.8 → 60.3 accuracy), while cross reflection yields 58.3. For gpt-3.5-turbo, cross-reflection with GPT-5 more than doubles average accuracy (16.1 → 33.9). **Gemma 3 12B** again behaves as an intermediate executor: optimized with GPT-5 lifts it from 25.8 to 46.9 average accuracy, competitive with the gpt-3.5-turbo variants but still below the gpt-4.1-mini. These are multiple-choice tasks, so the gains mostly reflect better calibration and more consistent mapping from internal reasoning to a single label.

Belebele reading comprehension (Table 4) reveals a complementary pattern. For gpt-4.1-mini, the base prompt already performs strongly (66.8 average accuracy), and GEPA provides only marginal additional gains (67.2 with self-reflection, 67.3 with cross-reflection). In contrast, gpt-3.5-turbo is extremely weak under the base prompt (3.8 average accuracy, with some languages below 1%). GEPA dramatically improves this, especially with cross reflection, to 31.8 average accuracy. Here, prompt optimization is effectively the difference between an unusable and a usable model for this task–language combination.

For PIQA-Yorùbá (Table 3), we again see moderate but consistent gains: gpt-4.1-mini improves from 66.0 to 72.0 with self-reflection, and gpt-3.5-turbo from 46.0 to 51.0 with GPT-5 reflection.

4.3 Machine Translation: AfriDocMT

On AfriDocMT *doc_health_5* and *doc_tech_5*, we evaluate ChrF++ for document-level translation. Table 5 shows consistent but more modest improvements. For gpt-4.1-mini, cross reflection raises Yorùbá En→Yo from 34.62 to 41.81 chrF++ and

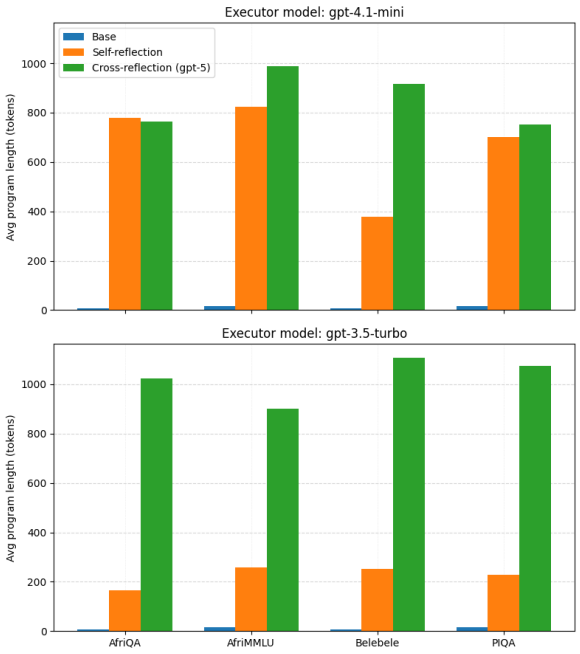

Figure 2: Average program length (tokens) per task and model variant. For each task, optimized entries report the mean token length of GEPA-generated programs averaged across that task's languages. Self-reflection uses the executor model as the reflection model; cross-reflection uses GPT-5.

Yo→En from 60.19 to 62.77; similar gains appear for Swahili and, to a lesser extent, Hausa. gpt-3.5-turbo also benefits (e.g., Swahili En→Sw: 61.93 → 68.57). Gemma 3 12B exhibits moderate but consistent gains as well: chrF++ improves by about 3–5 points across most directions under cross reflection with gpt-5. These results indicate that reflective prompt optimization is not limited to classification or span extraction, but can also improve structured generation when prompts explicitly encode formatting and segmentation instructions.

4.4 Latency–Quality Trade-off and Prompt Length

The arrow plots in Figure 3 and Figure 4 visualize the trade-off between prompt length and performance for AfriQA (gpt-4.1-mini, cross reflection) and Belebele (gpt-3.5-turbo, cross reflection). Each arrow starts at the base prompt (short, low-performance) and ends at the optimized prompt (longer, higher-performance). The vertical dashed line marks the average optimized prompt length for that setting, and the horizontal dashed line marks the corresponding average score.

For AfriQA, all languages move into the upper-right quadrant: EM improves by 10–20 points while program length typically grows from 8 tokens to hundreds (Figure 2). For Belebele with

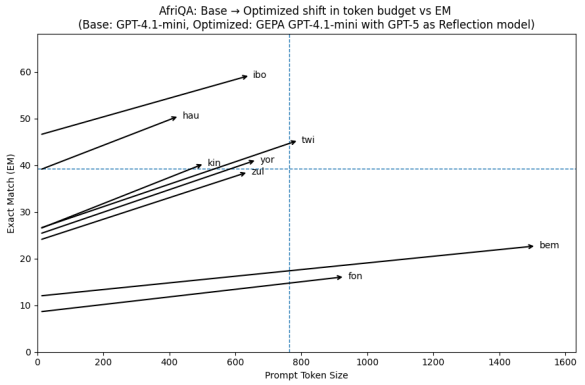

Figure 3: Token–quality trade-off induced by GEPA on AfriQA for GPT-4.1-mini under cross-model reflection (GPT-5). Each arrow represents one language, moving from the base to the optimized program in (prompt tokens, exact-match) space. Horizontal shifts indicate increases in prompt length, while vertical shifts indicate exact-match gains, revealing language-dependent efficiency in token utilization.

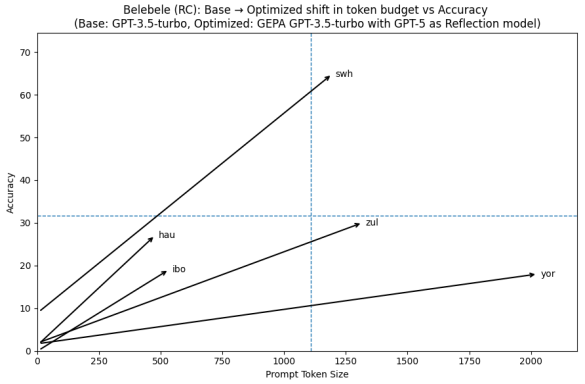

Figure 4: Token vs accuracy trade-off induced by GEPA on Belebele reading comprehension for GPT-3.5-turbo under cross-model reflection (GPT-5). Arrows show language-wise transitions from base to optimized prompts in (prompt tokens, accuracy) space. Horizontal shifts indicate increased prompt length, while vertical shifts indicate accuracy gains, with especially large improvements for Swahili and Zulu, highlighting the effectiveness of reflective prompt evolution for weaker base models.

gpt-3.5-turbo, the base prompt lies near the origin, and GEPA moves languages such as Swahili and Zulu to accuracies above 60% at the cost of 500–1500 prompt tokens. In both plots, languages with the weakest baselines often see the steepest arrows, highlighting that reflective optimization is especially valuable when naive prompting fails.

Figure 2 summarizes average program length across tasks and models. Base prompts are extremely compact (8–17 tokens), while optimized prompts are substantially longer. For gpt-4.1-mini, self-reflection yields an average of 670 tokens across tasks, and cross reflection about 855 tokens. For gpt-3.5-turbo, self-reflection averages 226 tokens, whereas cross-reflection pushes length

to around 1027 tokens. Thus, self-reflection is noticeably more parameter-efficient in terms of token budget, though cross-reflection often yields larger performance gains.

Token tables in Appendix C provide the full per-task, per-language breakdown. Overall, the results suggest a favorable trade-off in the low-resource setting: a one-time optimization phase and an increase from tens to a few hundred prompt tokens can unlock substantial accuracy and F1 gains, especially for weaker models and harder languages. In settings where latency or cost is critical, self-reflection may be preferred; where quality is paramount and token budget allows, cross-reflection with a stronger model offers further improvements.

4.5 Prompt Sensitivity and Cross-Lingual Transfer

The AfriQA experiments highlight both the severity of prompt sensitivity and the potential for cross-lingual sharing. Baseline EM varies widely across languages even for the same model (e.g., gpt-4.1-mini performs much better on Igbo and Hausa than on Fon or Bemba under the seed prompt). Prompt optimization narrows these gaps by converging towards a set of shared behaviors: tight control of answer span, explicit mapping from internal choices to output format, and language-agnostic instructions that avoid overfitting to any particular orthography.

The Twi-Optimized condition further shows that reflective optimization can produce prompts that generalize across languages without seeing them during optimization. This is encouraging for low-resource languages where development data are scarce or expensive: optimizing on one or two related languages may already yield robust prompts that transfer to others. We therefore argue that future benchmarks for African and other low-resource languages should report both baseline and prompt-optimized performance, under a clearly specified optimization budget, to more accurately reflect the capabilities of the underlying models.

Limitations

Our study has certain limitations. First, we only evaluate a small set of models (gpt-3.5-turbo, gpt-4.1-mini, gpt-5, and Gemma 3 12B). Extending these experiments to diverse open-weight models, and to additional African languages beyond those covered by AfriQA, AfriMMLU, Belebele, PIQA, and AfriDocMT, is important future work.

Second, we use a single GEPA configuration (`auto="light"`) with a fixed, relatively small optimization budget. This choice reflects realistic engineering constraints, but more aggressive optimization or alternative prompt optimizers might change the balance between prompt length, latency, and quality.

Finally, we treat the reflection model including in the self-reflection setting as an oracle for feedback. We do not model how its biases or failure modes might shape the learned prompts, especially for languages where the reflection model itself is weak. Understanding when reflective optimization amplifies dataset artifacts or cross-lingual biases is an important direction for future work.

References

Ife Adebara, Hawau Olamide Toyin, Nahom Tesfu Ghebremichael, AbdelRahim A. Elmadany, and Muhammad Abdul-Mageed. 2025. Where are we? evaluating LLM performance on African languages. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32704–32731, Vienna, Austria. Association for Computational Linguistics.

David Ifeoluwa Adelani, Jessica Ojo, Israel Abebe Azime, Jian Yun Zhuang, Jesujoba Oluwadara Alabi, Xuanli He, Millicent Ochieng, Sara Hooker, Andiswa Bukula, En-Shiun Annie Lee, Chiamaka Ijeoma Chukwuneke, Happy Buzaaba, Blessing Kudzaishe Sibanda, Godson Koffi Kalipe, Jonathan Mukiibi, Salomon Kabongo Kabenamualu, Foutse Yuehgoh, Mmasibidi Setaka, Lolwethu Ndolela, and 8 others. 2025. IrokoBench: A new benchmark for African languages in the age of large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2732–2757, Albuquerque, New Mexico. Association for Computational Linguistics.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *Preprint*, arXiv:2507.19457.

Benjamin Akera, Evelyn Nafula, Gilbert Yiga, Phionah Natukunda, Solomon Nsumba, Joel Muhanguzi, Janat Namara, Imran Sekalala, Patrick Walukagga, Engineer Bainomugisha, Ernest Mwebaze, and John

Quinn. 2025. Sunflower: A new approach to expanding coverage of african languages in large language models.

Jesujoba O. Alabi, Israel Abebe Azime, Miaoran Zhang, Cristina España-Bonet, Rachel Bawden, Dawei Zhu, David Ifeoluwa Adelani, Clement Oyeleke Odoje, Idris Akinade, Iffat Maab, Davis David, Shamsuddeen Hassan Muhammad, Neo Putini, David O. Ademuyiwa, Andrew Caines, and Dietrich Klakow. 2025. Afridoc-mt: Document-level mt corpus for african languages. *Preprint*, arXiv:2501.06374.

Sotiris Anagnostidis and Jannis Bulian. 2024. How susceptible are LLMs to influence in prompts? In *First Conference on Language Modeling*.

Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. 2024. The belebele benchmark: a parallel reading comprehension dataset in 122 language variants. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 749–775. Association for Computational Linguistics.

Tyler A. Chang, Catherine Arnett, Abdelrahman Eldesokey, Abdelrahman Sadallah, Abeer Kashar, Abolade Daud, Abosede Grace Olanihun, Adamu Labaran Mohammed, Adeyemi Praise, Adhikarinayum Meerajita Sharma, Aditi Gupta, Afitab Iyigun, Afonso Simplício, Ahmed Essouaied, Aicha Chorana, Akhil Eppa, Akintunde Oladipo, Akshay Ramesh, Aleksei Dorkin, and 319 others. 2025. Global piqa: Evaluating physical commonsense reasoning across 100+ languages and cultures. *Preprint*, arXiv:2510.24081.

Yuyan Chen, Zhihao Wen, Ge Fan, Zhengyu Chen, Wei Wu, Dayiheng Liu, Zhixu Li, Bang Liu, and Yanghua Xiao. 2024. Mapo: Boosting large language model performance with model-adaptive prompt optimization. *Preprint*, arXiv:2407.04118.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. 2026. Reinforcement learning via self-distillation. *Preprint*, arXiv:2601.20802.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. Dspy: Compiling declarative language model calls into self-improving pipelines.

Yinquan Lu, Wenhao Zhu, Lei Li, Yu Qiao, and Fei Yuan. 2024. LLaMAX: Scaling linguistic horizons of LLM by enhancing translation capabilities beyond 100 languages. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10748–10772, Miami, Florida, USA. Association for Computational Linguistics.

Odunayo Ogundepo, Tajuddeen R. Gwadabe, Clara E. Rivera, Jonathan H. Clark, Sebastian Ruder, David Ifeoluwa Adelani, Bonaventure F. P. Dossou, Abdou Aziz DIOP, Claytone Sikasote, Gilles Hacheme, Happy Buzaaba, Ignatius Ezeani, Rooweither Mabuya, Salomey Osei, Chris Emezue, Albert Njoroge Kahira, Shamsuddeen H. Muhammad, Akintunde Oladipo, Abraham Toluwase Owodunni, and 33 others. 2023. Afriqa: Cross-lingual open-retrieval question answering for african languages. *Preprint*, arXiv:2305.06897.

Jessica Ojo, Odunayo Ogundepo, Akintunde Oladipo, Kelechi Ogueji, Jimmy Lin, Pontus Stenetorp, and David Ifeoluwa Adelani. 2025. AfroBench: How good are large language models on African languages? In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19048–19095, Vienna, Austria. Association for Computational Linguistics.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. 2024. Optimizing instructions and demonstrations for multi-stage language model programs. *Preprint*, arXiv:2406.11695.

Amirhossein Razavi, Mina Soltangheis, Negar Arabzadeh, Sara Salamat, Morteza Zihayat, and Ebrahim Bagheri. 2025. Benchmarking prompt sensitivity in large language models. *Preprint*, arXiv:2502.06065.

Hao Yu, Tianyi Xu, Michael A. Hedderich, Wassim Hamidouche, Syed Waqas Zamir, and David Ifeoluwa Adelani. 2026. Afriquellm: How data mixing and model architecture impact continued pre-training for african languages. *Preprint*, arXiv:2601.06395.

Appendix

A Metrics and Feedback

A.1 Belebele (Multiple-Choice Reading Comprehension): Metric and Feedback

For ‘Belebele‘, a multiple-choice reading comprehension task. Each example provides a passage, a question, four answer options, and a gold option index. The metric we employed is exact-match accuracy on the predicted option index; feedback is constructed from three cases: invalid option, correct, or incorrect as shown below:

```
Inputs:
  question: string
  passage: string (optional)
  answer_list: [a1, a2, a3, a4]
  gold = correct_answer_number  {1,2,3,4}
  pred = prediction.correct_answer_number

Metric:
  if pred is not in {1,2,3,4}:
    score = 0
  else:
    score = 1 [pred = gold]

Feedback:
  let ag = answer_list[gold]

  if pred is not in {1,2,3,4}:
    feedback =
      "You are to return a single option number in [1, 2, 3, 4]. " +
      "You returned '{pred}', which is not a valid option. " +
      "The correct option is '{gold}', which corresponds to: {ag}."

  else if pred = gold:
    feedback =
      "Your answer is correct! It is option '{gold}', which corresponds to: {ag}."

  else:
    feedback =
      "Your answer is incorrect. The correct option is '{gold}', which corresponds to: {ag}."

  if passage is present:
    feedback +=
      "Here is the passage that contains the correct answer:" +
      "[passage]" +
      "Next time: locate the sentence that directly answers the question, then map it to the
     option list."

Return: (score, feedback)
```

A.2 AfriQA (Question Answering): Metric and Feedback

For `AfriQA`, a question answering task, each example provides a question and one or more gold short answers in an African language, alongside a supporting context passage in English. While both Exact Match (EM) and token-level F1 are used for evaluation before and after optimization, the reflective feedback signal is constructed using Exact Match.

```
Inputs:
  question: string
  context: string
  golds = answer_lang (list of short strings)
  pred = prediction.answer_lang

Metric (Exact Match):
  normalize(pred)
  normalize(golds)

  if pred matches any gold exactly:
    score = 1
  else:
    score = 0

Feedback:
  if pred is empty or missing:
    feedback =
      "The final answer must be a non-empty short string and nothing else. " +
      "You responded with '{pred}'."

    if golds are available:
      feedback +=
        " The correct exact-match answer is one of: {golds}."

    if context is present:
      feedback +=
        "Here's some context useful in answering the question:" +
        "[context]" +
        "Think about what takeaways you can learn from this context to improve your future
    answers "
        "and approach to similar question."

  else if pred exactly matches any gold:
    feedback =
      "Your answer is correct. The correct answer is '{pred}'."

  else:
    feedback =
      "Your answer is incorrect. The correct and exact-match answer is one of: {golds}."

    if context is present:
      feedback +=
        "Here's some context useful in answering the question:" +
        "[context]" +
        "Think about what takeaways you can learn from this context to improve your future
    answers "
        "and approach to similar question."

Return: (score, feedback)
```

A.3 AfriMMLU (Multiple-Choice QA): Metric and Feedback

For AfriMMLU, a multiple-choice question answering task. Each example provides a question, a subject label, four answer choices, and a gold answer letter in {A,B,C,D}. The metric we employed is exact-match accuracy on the predicted answer letter; feedback is constructed from three cases: invalid letter, correct, or incorrect as shown below:

```
Inputs:
  question: string
  subject: string
  choices: [cA, cB, cC, cD]
  gold = example.answer  {A,B,C,D}
  pred = prediction.answer

Metric:
  pred_letter = upper(strip(pred))

  if pred_letter is not in {A,B,C,D}:
    score = 0
  else:
    score = 1 [pred_letter = gold]

Feedback:
  letters = [A, B, C, D]
  correct_letter = gold
  correct_choice = choices[index(correct_letter)]

  pred_letter = upper(strip(pred))

  if pred_letter is not in {A,B,C,D}:
    feedback =
      "The final answer must be one of A, B, C, or D. " +
      "You responded with '{pred}', which is not a valid option. " +
      "The correct answer is '{correct_choice}', which is option {correct_letter}."

  else if pred_letter = correct_letter:
    feedback =
      "Your answer is correct. The correct answer is " +
      "'{correct_choice}', which is option {correct_letter}."

  else:
    feedback =
      "Your answer is incorrect. The correct answer is " +
      "'{correct_choice}', which is option {correct_letter}."

Return: (score, feedback)
```

A.4 PIQA (Physical Commonsense Reasoning): Metric and Feedback

For PIQA, a binary-choice physical commonsense reasoning task. Each example provides a goal, two candidate solutions, and a gold label in {0,1} indicating the correct solution. The metric we employed is exact-match accuracy on the predicted label; feedback is constructed from three cases: invalid label, correct, or incorrect as shown below:

```
Inputs:
  goal: string
  solutions: [s0, s1]
  gold = example.label  {0,1}
  pred = prediction.label

Metric:
  if pred is not in {0,1}:
    score = 0
  else:
    score = 1 [pred = gold]

Feedback:
  correct_idx = gold
  correct_solution = solutions[correct_idx]

  if pred is not in {0,1}:
    feedback =
      "The final answer must be either '0' or '1'. " +
      "You responded with '{pred}', which is not a valid option. " +
      "The correct label is '{gold}', which corresponds to: {correct_solution}"

  else if pred = gold:
    feedback =
      "Your answer is correct. The correct label is '{gold}', " +
      "which corresponds to solutions[{correct_idx}]: {correct_solution}"

  else:
    feedback =
      "Your answer is incorrect. The correct label is '{gold}', " +
      "which corresponds to solutions[{correct_idx}]: {correct_solution}"

Return: (score, feedback)
```

A.5 AfriDocMT (Machine Translation): Metric and Feedback

For `AfriDocMT`, a machine translation task. Each example provides a source sentence and a gold target translation. We compute chrF++ between the model translation and the reference; the resulting score is used both as the optimization metric and to select one of three feedback templates (`low`, `medium`, `high`) based on fixed thresholds, as shown below:

```
Inputs:
  source_text: string
  correct_translation = example.target_text
  model_translation = prediction.target_text

Metric (chrF++):
  score = chrF++(model_translation, correct_translation)
  (computed with word_order = 2 and lowercase = True)

Feedback:
  if score  0.2:
    feedback =
      "This translation is incorrect and contains significant errors. " +
      "The correct translation is \n {correct_translation} \n\n " +
      "Your translation has very low chrF++ which implies the output has almost no overlap " +
      "with the reference and fails to convey the source meaning."

  else if score < 0.5:
    feedback =
      "This translation is partially correct but contains some errors. " +
      "The correct translation is \n {correct_translation} \n\n " +
      "Your translation has a moderate chrF++ score which indicates that while some parts " +
      "of the translation are accurate, there are still noticeable discrepancies compared " +
      "to the reference."

  else:
    feedback =
      "This is a correct translation with minor or no errors. " +
      "The correct translation is \n {correct_translation} \n\n " +
      "Your translation has a high chrF++ score which indicates that it closely matches " +
      "the reference and effectively conveys the source meaning."

Return: (score, feedback)
```

B Additional Examples of GEPA Optimized Prompts

B.1 Belebele (Reading Comprehension)

BASE PROMPT

```
Answer a multiple-choice question using the passage
Question: $question,  Passage: $passage, Answer_list: $answer_list, Correct_answer_number:
```

OPTIMIZED PROMPT

```
Task
Select the correct option (1-4) for a multiple-choice question strictly using only the given passage (often in isiZulu). Return exactly one character: '1', '2', '3', or
'4'. No brackets, quotes, spaces, newlines, or explanations.

Inputs
- passage: Short text (may be isiZulu; may include quotations and reported speech).
- question: A query about the passage. It may ask for:
  - What is stated ("Ngokwendaba/Ngokusho kwesiqephu/Ngokombiko/Ngokwalokho okushiwo").
  - NOT mentioned/negation ("engavezwanga", "akushiwongo", "akukhulunywanga", "akuzange kushiwo/akukho okushiwo", "akunaso/awukho umbiko").
  - Should/must ("kumelwe/kumele", "kufanele").
  - Suggests/implication ("kusikisela", "kusho ukuthi", "kungenzeka").
  - Cause/source ("imbangela", "umsuka", "ngenxa ka/ye").
  - Who/where/when ("ubani" = who, "kuphi" = where, "nini" = when).
- answer_list: Four options in order. Their positions map to outputs '1'-'4'.
Output
- Exactly one character: '1', '2', '3', or '4'.

Method
1) Identify the question type and scope:
   - According to the passage: choose what the passage explicitly states.
   - NOT mentioned: choose what is absent or not supported by the passage.
   - Should/must: choose what the passage says must/should be done.
   - Suggests/implication: choose the best-supported implication.
   - Cause/source: choose the stated cause or source.
   - Who/where/when: pick the named agent/place/time asked.
2) Find the exact sentence(s) in the passage that answer the question.
   - Prefer explicit statements over plausibility or world knowledge.
   - For "kuphi" (where), pick the specific named place (e.g., "EsiGodini saseCanal").
   - For "ubani" (who), pick the actual agent/attributor (entity that "ithe/kusho/imenezele/kubika").
   - For "nini" (when), extract the stated time marker (e.g., "phakathi neSiphepho u-Katrina" = during Hurricane Katrina).
3) Handle reported speech carefully:
   - Track who said what via verbs like "ithe", "kusho", "imenezele", "kubika".
   - If A announces that B said/labelled "X", attribute "X" to B, not A.
4) NOT mentioned questions:
   - List mentally what is explicitly present or clearly implied.
   - Do not select items merely plausible from world knowledge.
   - If an option is more specific than the passage and that specificity does not appear, treat it as NOT mentioned.
5) Match paraphrases/synonyms and numeric details:
   - Negation of seriousness: "into engeyona enkulu", "akubucayi", "akunankinga enkulu".
   - Coffee terms: "short black" = "espresso"; "flat white" = coffee with steamed milk.
   - Place prefixes: "e-" marks location names (e.g., "eBuffalo").
   - Institutional markers: "i-Apple", "INkonzo yoMnotho… (METI)", "Umnyango wezicisha mlilo".
   - Translations/aliases:
     - "Uhamba Phambili" = "Number One/Front Runner".
     - "Iwodi Lesishiyagalolunye" = "Ninth Ward".
     - "Binnestad" refers to Amsterdam's inner city; "Grachtengordel/Canal District" is a specific Amsterdam area.
     - "amagxathu" and "izinyathelo" both mean steps; use numbers exactly (e.g., "amagxathu/izinyathelo ezingu-20").
     - "amayintshi angu-6" = six inches; "amafidi" = feet.
   - Prefer options that include the exact critical detail (e.g., specific location name or exact number) matching the question.
6) Tie-breakers:
   - Prefer explicit over inferred.
   - Match the exact scope of the question (who vs where vs when vs cause).
   - If two options seem true, choose the one that directly includes the key detail the question asks (e.g., the precise place name or the exact number mentioned).
7) Final checks:
   - Answer the exact question type (e.g., NOT mentioned vs stated; where vs what).
   - Ensure the chosen option text is supported by the located passage sentence(s).
   - Map to the option's index in answer_list (1-4).
   - Output only '1', '2', '3', or '4'.

Common pitfalls to avoid
- Do not output arrays, brackets, or explanations. Only a single digit.
- Do not choose a general or related fact if the question asks for a specific one (e.g., if asked "where in Amsterdam are house prices among the highest?", choose the
named area "EsiGodini saseCanal", not a generic "UNESCO World Heritage").
- Do not confuse cause with event or location scope. If asked what happened in the Ninth Ward during Katrina, choose the explicit depth there (e.g., "izinyathelo ezingu-
20"), not rainfall elsewhere ("amayintshi angu-6") or levee overflow facts unless the question is about the levee.
- Do not misattribute quotes or statements in reported speech.
- Do not use outside knowledge; rely solely on the passage.
Formatting rule (absolute)
- Return exactly one character: '1', '2', '3', or '4'.
```

Figure 5: Reflective optimized prompt for Belebele (Zulu). Using `gpt-5` as the reflection model, GEPA transforms a weak baseline prompt for `gpt-3.5-turbo`, raising accuracy from **2.0** to **30.0**.

B.2 AfriMMLU (Multiple-Choice QA)

BASE PROMPT

```
Answer the question with the correct option (A-D) from the given choices
Quesion: $question,  Subject: $subject, Choices: $choices, Answer:
```

↓

OPTIMIZED PROMPT

```
Task: Read a multiple-choice question and output only the single letter (A-D) that corresponds to the correct choice.

Input format:
- question: A natural-language question (may be in Akan/Twi or English).
- subject: A topical hint (e.g., global_facts, high_school_geography). Use only as context; it does not change the output format.
- choices: Exactly four options in order. Map letters as:
  A = choices[0], B = choices[1], C = choices[2], D = choices[3].

What to do:
1. Read and understand the question (mentally translate Twi to English if needed; never output the translation).
2. Use sound reasoning and elimination to select the best of the four choices.
3. Output ONLY the corresponding uppercase letter A, B, C, or D.

Strict output rules:
- Output exactly one character: A, B, C, or D.
- Do not include the option text, words, numbers, punctuation, brackets, quotes, or explanations.
- Do not output indices like [4] or the choice content (e.g., '11%'); only the letter (e.g., D).
- No spaces or newlines before/after the letter.

Reasoning and caution:
- Carefully map the chosen choice to its letter (A=1st, B=2nd, C=3rd, D=4th).
- Translate Twi nuances mentally; choose the semantically correct option.
- For quantitative/computational items, compute precisely before mapping.
- If uncertain, use logical elimination; never output multiple letters.
- Do not invent facts not implied by the question; avoid unrelated assumptions.

Known reference facts (for consistency across similar questions):
- Under-five deaths by country in 2017 (among choices China, United States, Indonesia, Pakistan): Pakistan is highest (corresponded to letter D in that set).
- Kinsey's 1948 report on male sexual behavior: the frequently cited statistic is 11% (corresponded to letter D in that set).
- High school geography in Twi: a scenario where students living in a school dorm eat at the nearest dining area exemplifies "Kwan ntam porɔɛɛ" (intervening opportunity)
(corresponded to letter B in that set).
- International law (ICJ Statute Article 38) in Twi: for 'amanaman mmara farebae', the correct choice was "Gye apam a wɔde di dwuma a kuo a ɔretwe manso no hyɛ asɛe pɛ"
(treaties in force) (corresponded to letter B in that set).
- South Africa's Grootboom case (Government of the Republic of South Africa v Grootboom): the Court required the state to take reasonable measures within available
resources to progressively realize the right of access to adequate housing, including provision for those in desperate need; no immediate individual entitlement to
housing. In a prior set, this corresponded to the option describing "reasonable measures" and "progressive realization" (letter C in that set).
- Central Place Theory (Walter Christaller): larger settlements provide a greater variety of functions/services; settlement size correlates with functional diversity. In
Twi, this was "Nkuro a ne kesɛyɛ ne no nnwumadie no yɛ ahodoɔ bebree" (letter C in that set).
- Detroit's population decline: commonly attributed to deindustrialization/decline of manufacturing jobs. In Twi, "Nnuwmaye a ɛbɛkɔ fam" (letter C in that set).

Final check before submitting:
- Confirm the best choice and its exact A-D mapping.
- Ensure the output is exactly one uppercase letter with no extra characters.
```

Figure 6: Reflective optimized prompt for AfriMMLU (Twi). GEPA optimizes a baseline multiple-choice reasoning prompt for `gpt-3.5-turbo` using `gpt-5` as the reflection model. This optimization increases accuracy from **8.8** to **32.0**, illustrating the impact of reflective prompt evolution on knowledge-intensive question answering.

B.3 PIQA (Physical Commonsense Reasoning)

BASE PROMPT

```
Pick the better solution (0 or 1) for the given goal.
Goal: $goal,  Solutions: $solutions, Label:
```

↓

OPTIMIZED PROMPT

```
You will be provided with a goal and two possible solutions. The goal and the solutions are always given in Yoruba language. Your task is to select which solution better
fulfills the stated goal by leveraging deep knowledge of Yoruba language, culture, customs, and domain-specific expertise.

Instructions

1. Understand the Goal in Yoruba:
   - Read the Yoruba goal carefully.
   - Identify the cultural, contextual, or domain category it belongs to (e.g., traditional objects, ecological knowledge, Yoruba terminology, customary practices, Yoruba
proverbs or idioms, ritualistic expressions, indigenous tools or crafts, or local natural phenomena).
   - Recognize that many goals require understanding beyond literal translation: they hinge on knowing specific Yoruba cultural facts or terminologies accurately.

2. Analyze Both Yoruba Solutions:
   - Examine each solution's wording and implied meaning using culturally informed knowledge.
   - Determine which solution aligns with Yoruba traditional use, linguistic nuance, cultural appropriateness, or factual correctness.
   - Discard solutions that:
     * Misuse Yoruba cultural terms or objects,
     * Present inaccurate cultural or ecological facts (e.g., wrong habitat for animals/plant eggs),
     * Misapply traditional language or customary expressions (e.g., incorrect greeting forms for royalty, improper use of Yoruba hunting or farming terms),
     * Reflect literal but culturally incorrect interpretations.

3. Apply Relevant Yoruba Cultural & Domain-Specific Knowledge. Examples include:
   - Traditional greetings to Yoruba kings must use culturally respectful phrases such as "Kábiyèsí o" rather than generic greetings.
   - Knowledge of Yoruba indigenous tools or crafts, such as understanding what a "jẹnẹrátọ̀" (generator) is versus an "ọ̀kadà" (a farming or manual tool).
   - Indigenous fishing or trapping methods and materials: for instance, "okùn gígùn" (long rope) may function better than "ówú" (cotton/thread) in making fishing traps.
   - Understanding Yoruba agricultural or natural product processing: e.g., "amọ̀" (clay) is traditionally used to preserve or shape "kòkó" (a local product), while "àpò"
(bag) is not.
   - Clear identification of habitat or ecological sites in Yoruba for specific items (like where salmon eggs are found or hunting-related terms).
   - Precise definitions of Yoruba lexical items like "ṣibi" referring to locations or functional hunting places, not food.

4. Reason Your Choice:
   - Provide a short but culturally informed explanation citing specific Yoruba customs, terms, or factual knowledge supporting your choice.
   - Point out why the rejected option is culturally or factually incorrect or less appropriate.

5. Output:
   - Your final answer must be the number label (0 or 1) corresponding to the solution that better satisfies the goal.

Summary of Effective Approach:
- Identify the cultural or domain context of the goal.
- Link each solution to that contextual knowledge.
- Reject solutions displaying cultural, ecological, or linguistic inaccuracies.
- Select the solution best reflecting authentic Yoruba knowledge and culturally accurate practice.

This method ensures accurate answers grounded in deep Yoruba language and cultural understanding, beyond surface literal translations.
```

Figure 7: Self-reflective optimized prompt for PIQA (Yorùbá). GEPA refines the baseline prompt for `gpt-4.1-mini` using the same model as the reflection model. This self-reflection improves accuracy from **66.0** to **72.0**, demonstrating that meaningful gains are possible even without a stronger reflection model.

C Prompts Token Sizes

C.1 AfriQA Programs

			Prompt Tokens							
Model	**Setting**	**Reflection Model**	Yor	Hau	Ibo	Zul	Fon	Bem	Kin	Twi
gpt-4.1-mini	Base	–	8	8	8	8	8	8	8	8
gpt-4.1-mini	Optimized	gpt-4.1-mini	921	890	536	843	987	557	519	989
gpt-4.1-mini	Optimized	gpt-5	664	429	644	638	931	1511	505	791
gpt-3.5-turbo	Base	–	8	8	8	8	8	8	8	8
gpt-3.5-turbo	Optimized	gpt-3.5-turbo	232	163	111	139	40	235	355	42
gpt-3.5-turbo	Optimized	gpt-5	1245	1162	1619	790	1077	1195	608	505
gpt-3.5-turbo	Twi-Optimized	gpt-5	505	505	505	505	505	505	505	–
gpt-4.1-mini	Twi-Optimized	gpt-5	791	791	791	791	791	791	791	–
gpt-5	Base	–	8	8	8	8	8	8	8	8

Table 6: Prompt token lengths for AfriQA programs. For the ‘Optimized‘ setting, each cell reports the token length of the language-specific optimized program. For ‘Twi-Optimized‘ setting, a single Twi-optimized program is reused across languages (hence constant values across columns).

C.2 AfriMMLU Programs

			Prompt Tokens								
Model	**Setting**	**Reflection Model**	Yor	Hau	Swa	Twi	Ibo	Kin	Wol	Lin	Avg
gpt-4.1-mini	Base	–	17	17	17	17	17	17	17	17	17
gpt-4.1-mini	Optimized	gpt-4.1-mini	414	433	2019	932	1053	925	391	421	823.5
gpt-4.1-mini	Optimized	gpt-5	792	992	1179	688	1768	853	582	1054	988.5
gpt-3.5-turbo	Base	–	17	17	17	17	17	17	17	17	17
gpt-3.5-turbo	Optimized	gpt-3.5-turbo	285	177	63	334	296	402	198	298	256.6
gpt-3.5-turbo	Optimized	gpt-5	1616	1006	651	808	698	775	1130	529	901.6
gpt-5	Base	–	17	17	17	17	17	17	17	17	17

Table 7: Program token lengths for AfriMMLU runs. Base programs use a fixed 17-token program; optimized programs vary by language depending on the GEPA-generated program.

C.3 Belebele Programs

			Prompt Tokens					Avg
Model	**Setting**	**Reflection Model**	Yor	Hau	Ibo	Zul	Swa	
gpt-4.1-mini	Base	–	8	8	8	8	8	8
gpt-4.1-mini	Optimized	gpt-4.1-mini	71	217	592	632	–	–
gpt-4.1-mini	Optimized	gpt-5	857	1159	722	1389	455	916.4
gpt-3.5-turbo	Base	–	8	8	8	8	8	8
gpt-3.5-turbo	Optimized	gpt-3.5-turbo	209	232	316	345	159	252.2
gpt-3.5-turbo	Optimized	gpt-5	2025	476	532	1316	1193	1108.4
gpt-5	Base	–	8	8	8	8	8	8

Table 8: Program token lengths for Belebele runs. Base programs use a fixed 8-token program; optimized programs vary by language depending on the GEPA-generated program.

C.4 PIQA-Yoruba Programs

Model	Setting	Reflection Model	Prompt Tokens (Yor)
gpt-4.1-mini	Base	–	15
gpt-4.1-mini	Optimized	gpt-4.1-mini	700
gpt-4.1-mini	Optimized	gpt-5	752
gpt-3.5-turbo	Base	–	15
gpt-3.5-turbo	Optimized	gpt-3.5-turbo	229
gpt-3.5-turbo	Optimized	gpt-5	1074
gpt-5	Base	–	15

Table 9: Program token lengths for PIQA-Yoruba runs. Base programs use a fixed 15-token program; optimized programs reflect the GEPA-generated program length.