To Write or to Automate Linguistic Prompts, That Is the Question

Marina Sánchez-Torrón **Daria Akselrod*** **Jason Rauchwerk***
Smartling
244 Fifth Avenue Suite 1471 New York, NY 10001
{msancheztorron,dakselrod,jrauchwerk}@smartling.com

Abstract

LLM performance is highly sensitive to prompt design, yet whether automatic prompt optimization can replace expert prompt engineering in linguistic tasks remains unexplored. We present the first systematic comparison of hand-crafted zero-shot expert prompts, base DSPy signatures, and GEPA-optimized DSPy signatures across terminology insertion, translation and language quality assessment, evaluating five model configurations. Results are task-dependent. For terminology insertion and translation, GEPA-optimized prompts are competitive with expert prompts: most differences are not statistically significant, and optimization significantly improves glossary term match rates for several models. In language quality assessment, expert prompts achieve stronger error detection while optimization improves characterization. Across tasks, GEPA elevates minimal DSPy signatures, often closing the gap to expert performance. We note that the comparison is asymmetric: GEPA optimization searches programmatically over gold-standard splits, whereas expert prompts require in principle no labeled data, relying instead on domain expertise and iterative refinement.

*Equal contribution.

1 Introduction

LLMs have shown remarkable capabilities across a wide range of NLP tasks. However, research shows that LLM performance is highly sensitive to the way prompts are worded and structured. Sclar et al. (2024) found that open-source LLMs exhibit significant performance differences when tested under slight, meaning-preserving variations in prompt formatting, such as adding or omitting punctuation marks; Voronov et al. (2024) found that variations in how examples are selected, verbalized or separated in few-shot settings results in performance differences, while Mizrahi et al. (2024) found that both manual and automated prompt paraphrasing lead to drastic performance variations in both open-source and closed models. While metrics have been proposed to evaluate this fragility (Errica et al., 2025; Sclar et al., 2024), improving prompts in response to such diagnoses remains a largely manual, iterative process requiring task-specific knowledge. This practice, referred to as prompt engineering, is the subject of a substantial body of research, producing for example catalogs of prompt patterns (White et al., 2023) and prompting techniques (Sahoo et al., 2025).

At scale, prompt engineering presents practical challenges. First, the process is inherently trial-and-error: practitioners iteratively test and revise prompts with no systematic way to determine whether their current prompt is optimal. Second, effective prompts do not necessarily transfer across setups: a prompt tuned for one model, task, or language pair may underperform when any of these changes, and model updates can degrade previously effective prompts.

These limitations have motivated research on automatic prompt optimization, progressing from

continuous gradient-based methods to discrete approaches (see Section 2). Recent gradient-free discrete optimizers (Yang et al., 2024; Guo et al., 2025; Agrawal et al., 2025) are directly applicable to the black-box API models on which LSPs typically rely, yet whether they can match expert-crafted prompts on specialized linguistic tasks remains unexplored.

We investigate this question across three linguistic tasks where language service providers (LSPs) increasingly rely on LLMs and where prompt design directly affects production quality: (a) terminology insertion, where an existing translation is corrected to conform to a provided glossary; (b) translation; and (c) language quality assessment (LQA), where translation errors are detected and characterized using an MQM-compliant (Lommel et al., 2014) schema.

This paper contributes to the current body of research by providing, to the best of our knowledge, the first comparison of expert-crafted prompts vs. optimized prompts for specialized linguistic tasks, with signatures and results available on GitHub *

2 Related Work

Automatic Prompt Optimization. The well-documented fragility of LLM performance under prompt variation (Sclar et al., 2024; Voronov et al., 2024; Mizrahi et al., 2024) has motivated a growing body of work on automatic prompt optimization. Early approaches treated the prompt as a learnable vector in the model's embedding space, optimizing it via gradient-based (Li and Liang, 2021; Liu et al., 2022) or reinforcement learning (Zhang et al., 2022b) techniques. These methods require access to model gradients, ruling out black-box API models, and produce continuous representations that are not interpretable as natural language. Discrete approaches operate directly over natural language tokens, producing human-readable prompts. Earlier discrete methods still relied on gradient access (Shin et al., 2020; Shi et al., 2022; Deng et al., 2022; Zhang et al., 2022a; Pryzant et al., 2023), but recent work has shown that effective discrete optimization is possible without access to model parameters (Yang et al., 2024; Guo et al., 2025; Agrawal et al., 2025), making automatic optimization directly applicable to the black-box API models on which LSPs typically rely.

Despite these advances, research directly comparing automatically optimized prompts against expert-crafted prompts remains limited. Existing studies demonstrate the benefits of prompt optimization but typically benchmark against generic baselines rather than domain-expert prompts, and focus on coding, classification, and question-answering rather than specialized linguistic tasks. For example, Agrawal et al. (2025) show that their GEPA optimizer outperforms Group Relative Policy Optimization (GRPO) and other optimizers, while Yang et al. (2024) and Yuksekgonul et al. (2024) show improvements over human-designed prompts without reference to the authors' expertise. Conversely, Zhou et al. (2023) found that simple manual prompts outperformed gradient-based automated prompting in the majority of setups, and He et al. (2025) found that both iterative human prompting and automated optimization within the DSPy framework (Khattab et al., 2023) were unreliable when gold-standard labels were scarce. Critically, none of these comparisons evaluate domain-expert prompts developed by professional practitioners, nor do they address specialized linguistic tasks where prompt quality has direct production consequences.

LLMs for Linguistic Tasks. LLMs have progressed rapidly on machine translation: early evaluations showed competitive performance on high-resource European languages but gaps in low-resource settings (Jiao et al., 2023; Hendy et al., 2023), with GPT-4 surpassing supervised baselines such as NLLB (NLLB Team et al., 2022) in many directions while still lagging behind commercial systems in low-resource languages (Zhu et al., 2024). More recent benchmarks show LLMs now dominate, with Gemini 2.5 Pro placing in the top evaluation cluster for 14 of 16 language pairs (Kocmi et al., 2025). Fine-grained human evaluations suggest LLM translations remain comparable to junior and mid-level professional translators but fall short of senior professionals (Yan et al., 2024).

For quality assessment, neural metrics such as COMET (Rei et al., 2020) and BLEURT (Sellam et al., 2020) correlate well with human assessment (Freitag et al., 2021), and xCOMET (Guerreiro et al., 2023) extends this with error span detection, but these learned metrics lack the explain-

*https://github.com/msancheztorron-smartling/EAMT2026submission

ability of prompted LLM evaluators. Kocmi and Federmann (2023b) showed that GPT-based models achieved state-of-the-art system-level correlation with human judgments, and their GEMBA-MQM metric (Kocmi and Federmann, 2023a) extended this to fine-grained error span detection. However, Fernandes et al. (2023) and Huang et al. (2024) found that predicted error spans do not align well with human annotations at the segment level, motivating refinements such as filtering LLM-predicted errors through automatic post-editing (Lu et al., 2025).

For terminology enforcement, treating terminology insertion as a post-translation step is closest to our setting. Moslem et al. (2023) showed that LLM-based post-editing can approximately double successful terminology insertion rates, while Kim et al. (2024) proposed a training-based approach using Trie-tree term extraction for specialized domains.

For all three tasks, prompt quality is known to matter, yet no prior work has compared automatically optimized prompts against domain-expert prompts for specialized linguistic workflows.

3 Methodology

3.1 Shared Experimental Setup

Across all tasks, we compare three prompting strategies: (a) hand-crafted zero-shot expert prompts; (b) base DSPy signatures without optimization; and (c) DSPy signatures optimized with GEPA (Agrawal et al., 2025). Manual and base DSPy configurations use language-agnostic prompts across all locales, with the target locale provided as an input variable. GEPA-optimized systems also use language-agnostic prompts, except for LQA, where optimization runs independently per language pair, producing locale-specific prompts. For the terminology insertion and LQA tasks, we compare unified single-stage systems against task-decomposed multi-stage pipelines, and evaluate all configurations with various models. All DSPy configurations interact with the LLMs through the `dspy.Predict` module.

Models. We evaluate five model configurations spanning four providers: (1) GPT-4.1-mini with GPT-4.1 for reflection; (2) GPT-5.4-mini with GPT-5.4 for reflection; (3) Gemini 3.1 Flash-Lite with Gemini 3.1 Pro for reflection; (4) Claude Sonnet 4.6 with Claude Opus 4.6 for reflection; and (5) Qwen3:8B-q4_K_M with Qwen3:14B-q4_K_M for reflection, hosted locally via Ollama. All configurations use temperature 0 and max 4K tokens for execution, and temperature 1 and 32K tokens for reflection.

Data. For all tasks, we use proprietary datasets spanning various domains, with gold-standard annotations produced by professional contract linguists. Annotations are reviewed internally, with in-house feedback used to refine and correct them before inclusion in the dataset. Data is split into train (20%), validation (40%), and test (40%) sets, as in Agrawal et al. (2025), who allocate the majority of data to validation and test since GEPA's reflective optimization requires relatively few training examples. Test set sizes are 1085 examples for terminology insertion, 989 for translation, and 1250 for LQA. Per-language results are omitted for space reasons but can be found in the accompanying GitHub repository.

Optimization. For GEPA optimization, we use "light" mode, which minimizes the number of programs proposed and evaluated per run. We choose this setting for two reasons: it reflects the practical constraint of optimizing across three tasks, five models, and multiple objectives within a manageable compute budget; and it establishes a conservative lower bound on optimization performance that provides a fairer comparison against zero-shot baselines. Textual feedback comparing gold vs. predicted outputs is provided to the optimizer in all configurations, with task-specific optimization objectives described in each task's subsection.

3.2 Terminology Insertion

Given a source text, its machine translation, and relevant glossary terms, our terminology insertion system performs automatic post-editing to enforce glossary compliance in translations where required terms are missing. If glossary violations are found, the system produces a mapping of incorrect terms in the original translation to the source terms in the glossary that is used for glossary lookup as a reasoning reference.

We experiment with terminology insertion in translations from English to six target locales (Arabic, Spanish, German, Russian, Traditional Chinese, Swedish). We follow the shared experimental setup (Section 3.1), with two task-specific details: task decomposition compares a unified glossary term insertion system against a three-stage

pipeline of violation identification, correction, and validation; and GEPA optimization targets three distinct objectives, detailed below.

Each example consists of a source text, original translation with approximately 70% missing glossary terms, source and target locale codes, and a dictionary of glossary terms previously automatically determined, via stemming, as present in the source text, but missing from the machine translated text. The relevance of glossary terms varies.[†] For example, the source text *"Start with lexical search"* contains *"search"* as a noun, but the detected glossary term for de-DE is a verb: `['sourceTerm': 'search', 'targetTerm': 'suchen']`. Although not relevant in context, this glossary is still given to the prompt. The manual prompts and DSPY signatures account for this uncertainty, instructing the model to disregard glossary entries that are incorrect or irrelevant and delegating the filtering decision to the LLM.

Optimization objectives. We optimize for three metrics: (1) BLEU with textual feedback comparing gold vs. predicted glossary-integrated translations and detailing missing or incorrectly inserted glossary terms by comparing stems; (2) HTER with textual feedback; (3) LLM-as-a-judge, using the reflection model, scoring glossary-integrated translations based on correct glossary usage, fluency, no extra edits, and correct capitalization.

Evaluation. We report BLEU, HTER (both using the SacreBLEU implementation) as discussed in Post (2018), measuring translation quality and edit distance from gold human translations, respectively, on the predicted glossary-integrated translation. We additionally report TMR (term match rate) as the rate at which glossary term stems that are present in the gold translation appear in the predicted translation. Both unified and task-decomposed systems are judged on the final produced glossary-compliant translation.

3.3 Translation

Given the source locale, target locale, source text, glossary, similar translated example text, and a translation style guide, our translation system produces output text in the target language that adheres to the stylistic constraints.

Our translation dataset spans ten target locales (German, Spanish, French, Italian, Japanese, Korean, Polish, Russian, Turkish, and Chinese), each translating from English.

Each example in the test dataset contains a source locale, a target locale, and a source text. Data points may contain any combination of glossary, similar translated text, and style guide, or none at all. This mimics the variety of linguistic resources available in real-world translation settings.

Optimization objectives. We optimize for the average between HTER and chrF3, as defined by Popović et al (2015). We use the SacreBLEU implementation for both scores. Together, these metrics provide a strong comparison between the machine-translated text and human-translated reference. To normalize both scores to the same scale, we use the formula $score = \frac{100-HTER}{200} + \frac{chrF3}{100}$

Evaluation. We report BLEU, HTER, and chrF3 scores calculated between the LLM output text and the gold-standard reference text. The mean and median scores are produced for the micro- and macro-average across the locales.

3.4 Language Quality Assessment

Given a source sentence and its machine translation, our LQA system predicts whether a translation error exists and, if so, its type and severity using an MQM-compliant schema of 15 error types and 3 severity levels.[‡]

We investigate the LQA task for translations from English into six target languages (Arabic, Spanish, German, Russian, Simplified Chinese, Swedish). Task decomposition compares unified error detection and characterization against a two-stage detection-then-characterization pipeline.

Each data instance consists of source text, target translation, and locale code as inputs, with binary error presence and a JSON array of errors, specifying category, severity, and annotator comment, as labels. Multiple errors per instance are supported. Error-bearing examples naturally constitute the majority of the data (∼65% across training, validation, and test splits).

[†]Part-of-speech is not included, glossaries may contain duplications, variants of target terms could be mapped to the same source term, and capitalization of terms is not always consistent.

[‡]Error types: Mistranslation, Omission, Addition, Grammar, Spelling, Punctuation, Unidiomatic, Register, Inconsistency, Language Variety, Whitespace, Markup/Technical, Locale Formatting, Duplication, Culture-specific reference; Severities: Minor, Major, Critical

Optimization objectives. Single-stage systems optimize a composite metric: 0.0 for detection mismatches, 1.0 for correct clean predictions, and $0.5 + 0.25 \times$ Category F1 $+ 0.25 \times$ Severity F1 for true positives. Given the class distribution, the optimizer cannot gain by defaulting to predicting no error at the example level. Two-stage systems optimize detection with a binary accuracy metric, while characterization (applied only to error-bearing examples) optimizes a hybrid score: $0.5 \times$ Category F1 $+ 0.5 \times$ Severity F1.

Evaluation. We report four primary metrics: Detection F1, Category F1, Severity F1, and Pearson's correlation with gold MQM scores.[§] All metrics are computed end-to-end on system outputs. Cross-locale summary statistics are macro-averaged, weighting each locale equally regardless of test set size.

4 Results

4.1 Terminology Insertion

Table 1 presents the results across all configurations. We highlight the key findings below.

Manual and optimized prompts achieve largely comparable terminology insertion quality across models. Manual prompts achieve BLEU scores higher than or equal to base DSPy for most models, with Qwen3:8B-q4_K_M as an exception where Base DSPy matches or exceeds Manual. Compared to optimized DSPy configurations, Manual BLEU scores are slightly higher or equivalent (1–2 point differences), though for Claude Sonnet 4.6 and GPT-4.1-mini decomposed, optimized prompts match or marginally surpass Manual. HTER differences are slightly more pronounced but remain non statistically significant: unified manual prompts achieve lower HTER than optimized DSPy for three of five models (GPT-4.1-mini, GPT-5.4-mini, and Gemini 3.1 Flash-Lite), with gaps of 2–4 points, while Claude Sonnet 4.6 shows a marginally lower HTER for optimized DSPy and Qwen3:8B-q4_K_M favors DSPy configurations more broadly. Optimized prompts achieved the highest term match rates, with statistically significant gains for half of the models, reflecting that optimization prioritizes glossary term presence over fluency as measured by HTER or BLEU.

Optimization objective has limited impact for most models, with notable exceptions. Across three of five models (GPT-4.1-mini, Gemini 3.1 Flash-Lite, and Qwen3:8B-q4_K_M), the three optimization objectives produced near-identical results. For GPT-5.4-mini and Claude Sonnet 4.6, the choice of objective did affect BLEU and HTER, with LLM-as-a-judge yielding the best scores. However, metric-aligned optimization does not guarantee metric-specific gains: optimizing against BLEU did not consistently produce the highest BLEU scores, nor did optimizing against HTER consistently minimize edit distance. In all cases, differences across objectives within a model were smaller than differences across models, with Gemini 3.1 Flash-Lite performing best overall regardless of optimization objective.

4.2 Translation

Table 2 presents the results for the translation task with the three metrics BLEU, HTER, and ChrF3. We highlight the key findings below.

Manual prompts generally win for newer models. On an older model like OpenAI's GPT-4.1-mini, DSPy prompts consistently win on all metrics. However, some modern models (GPT-5.4-mini and Gemini 3.1 Flash-Lite) actually perform best with manually crafted prompts. Claude Sonnet 4.6 is an exception.

Alternative prompting strategies perform statistically close to the best strategy. Although one strategy tends to dominate for a given model, it is typically not significantly better than the second-best alternative. Gemini 3.1 Flash-Lite is an exception: the manual prompt significantly outperforms Opt. DSPy on BLEU mean ($p < 0.01$). For all other models, alternative strategies can be substituted with minimal performance loss.

The open-weight model performed significantly worse than any proprietary model on translation. For Qwen3:8B-q4_K_M, generated prompts provide a statistically confirmed benefit: Base DSPy significantly outperforms the manual prompt on BLEU mean. However, even with this improvement, the model falls far short of every proprietary model across all metrics and strategies.

[§]MQM is calculated by weighting errors by severity (Minor=1, Major=5, Critical=25), normalizing by sentence length, and subtracting from a perfect base score of 100.

Model	Prompt	BLEU		HTER		TMR	
		Uni	Dec	Uni	Dec	Uni	Dec
GPT-4.1-mini	Manual	**0.48**	0.46	**0.53**	**0.56**	0.86	0.77
	Base	0.45	0.45	0.55	0.57	0.87	0.81
	Opt. DSPy (best)	$0.46^{(2,3)}$	$\mathbf{0.49}^{(1)}$	$0.55^{(3)}$	$\mathbf{0.56}^{(1,3)}$	$\mathbf{0.88}^{(1)}$	$\mathbf{0.91}^{(2)\ddagger}$
GPT-5.4-mini	Manual	**0.50**	**0.48**	**0.50**	**0.55**	0.90	0.82
	Base DSPy	0.45	0.45	0.56	0.57	0.85	0.78
	Opt. DSPy (best)	$0.49^{(3)}$	$0.47^{(1,2)}$	$0.52^{(3)}$	$\mathbf{0.55}^{(2)}$	$\mathbf{0.91}^{(3)}$	$\mathbf{0.91}^{(3)\ddagger}$
Gemini 3.1 Flash-Lite	Manual	**0.52**	**0.51**	**0.47**	**0.50**	**0.94**	0.92
	Base DSPy	0.50	0.50	0.51	**0.50**	0.92	0.91
	Opt. DSPy (best)	$0.50^{(1,2,3)}$	$\mathbf{0.51}^{(3)}$	$0.51^{(1,2,3)}$	$0.52^{(2,3)}$	$0.92^{(1,2,3)}$	$\mathbf{0.94}^{(2)}$
Claude Sonnet 4.6	Manual	0.49	0.47	0.51	0.55	0.90	0.84
	Base DSPy	0.48	0.47	0.54	0.55	**0.93**	**0.93**
	Opt. DSPy (best)	$\mathbf{0.50}^{(3)}$	$\mathbf{0.50}^{(3)}$	$\mathbf{0.50}^{(3)}$	$\mathbf{0.53}^{(3)}$	$\mathbf{0.93}^{(3)}$	$\mathbf{0.93}^{(3)}$
Qwen3:8B-q4_K_M	Manual	0.38	0.38	0.63	0.63	0.74	0.74
	Base DSPy	0.38	**0.40**	**0.59**	**0.61**	0.74	0.76
	Opt. DSPy (best)	$0.38^{(1,2,3)}$	$\mathbf{0.40}^{(2)}$	$\mathbf{0.59}^{(1,2,3)}$	$0.62^{(1,2)}$	$0.74^{(1,2,3)}$	$\mathbf{0.82}^{(1)\ddagger}$

Table 1: Summary performance for the Terminology Insertion task, macro-averaged across six target locales (ar, de-DE, es-ES, ru-RU, sv-SE, zh-TW). Uni = unified (single-stage); Dec = decomposed (three-stage) N: 1085 (ar=85, de-DE=200, es-ES=200, ru-RU=200, sv-SE=200, zh-TW=200). Each Opt. DSPy cell reports the score from the best-performing optimization objective for that metric and architecture; superscripts identify which:$^{(1)}$ BLEU, $^{(2)}$ HTER, $^{(3)}$ LLM-based judge. Best value per model and column in **bold**. ‡ $p < 0.01$ vs. the next-best configuration (paired stratified bootstrap, 1,000 iterations, Holm–Bonferroni corrected per model).

The best prompting strategy for each model is generally consistent across scores. The prompts were optimized against a mixture of HTER and ChrF3, but the best BLEU scores also coincide with the best overall prompt. This lends cross-metric confidence to the quality of the generated translations.

4.3 Language Quality Assessment

Table 3 presents the results across all configurations. We highlight the key findings below.

GEPA optimization consistently improves base DSPy signatures across model families. In all models, GEPA-optimized unified prompts match or exceed manual prompt performance on at least one primary metric. The pattern is most pronounced for characterization: decomposed optimized prompts achieve the highest Category F1 in four of five models, with Severity F1 showing a similar pattern.

Manual expert prompts retain an advantage on error detection. Across all five models, manual unified prompts achieve the highest Detection F1. However, this advantage over optimized prompts is statistically significant in only three of five models. In decomposed systems, optimized prompts significantly outperform manual prompts on Detection F1 in all models except Qwen3:8B-q4_K_M.

Unified systems outperform decomposed pipelines across all models. This finding generalizes robustly: in all five models, unified systems outperform their two-stage counterparts on Detection F1, typically by 0.05–0.20 points. We note that our significance tests compare prompt strategies within each architecture and do not directly test the unified-versus-decomposed comparison.

5 Conclusions

We presented, to our knowledge, the first systematic comparison of expert-crafted prompts against automatically optimized prompts for specialized linguistic tasks, evaluating across terminology insertion, translation, and LQA with five model configurations from four providers.

Central findings. GEPA optimization elevates minimal DSPy signatures, often closing the gap to expert-level performance, but the degree and direction of remaining differences are task-dependent. In terminology insertion, expert and optimized prompts are largely statistically indistinguishable on translation quality (BLEU, HTER), with optimization producing significantly higher term match rates for several models but not consistently improving fluency. In translation, most model–prompt differences are not statistically significant, with the exception of Gemini 3.1 Flash-Lite, where expert prompts significantly outperform optimized ones on BLEU. In LQA, optimization yields statistically significant gains over manual prompts on error characterization for several

Model	Prompt	BLEU		HTER		ChrF3	
		Mean	Median	Mean	Median	Mean	Median
GPT-4.1-mini	Manual	57.93	58.77	30.32	25.73	70.55	70.91
	Base DSPy	58.21	**60.15**	30.27	**25.41**	70.59	71.78
	Opt. DSPy	**59.33**	60.13	**29.21**	25.79	**71.27**	**72.64**
GPT-5.4-mini	Manual	**57.63**	**57.82**	**31.80**	**28.03**	**69.69**	**69.79**
	Base DSPy	55.60	55.19	34.07	29.55	68.88	69.06
	Opt. DSPy	56.16	55.81	33.16	29.26	69.18	69.41
Gemini 3.1 Flash-Lite	Manual	**60.75**‡	**61.31**	**29.20**	**25.04**	**72.61**	**73.45**
	Base DSPy	58.43	58.51	30.53	26.80	71.16	72.20
	Opt. DSPy	58.53	59.40	30.00	25.60	71.47	72.57
Claude Sonnet 4.6	Manual	58.81	59.32	29.49	25.21	71.54	72.97
	Base DSPy	58.42	58.99	30.31	25.80	71.23	72.24
	Opt. DSPy	**59.13**	**60.97**	**29.47**	**24.65**	**71.86**	**73.53**
Qwen3:8B-q4_K_M	Manual	46.06	42.74	40.41	39.16	61.53	60.64
	Base DSPy	**48.03**	44.37	**38.88**	37.14	**62.18**	61.29
	Opt. DSPy	47.33	**44.81**	39.44	**36.92**	61.98	**61.76**

Table 2: Summary performance for the Translation task, macro-averaged translation scores across ten target locales. N: 972 (de-DE=140, es-ES=69, fr-FR=133, it-IT=15, ja-JP=164, ko-KR=170, pl-PL=60, ru-RU=50, tr-TR=44, zh-CN=127). Mean and median of BLEU, HTER, and ChrF3. BLEU, ChrF3 = higher is better, HTER = lower is better. Best value per model and column in **bold**. † $p < 0.05$, ‡ $p < 0.01$ vs. the next-best configuration (paired stratified bootstrap, 1,000 iterations, Holm–Bonferroni corrected per model).

Model	Prompt	Det F1		Cat F1		Sev F1		MQM r	
		Uni	Dec	Uni	Dec	Uni	Dec	Uni	Dec
GPT-4.1-mini	Manual	**0.64**	0.54	0.36	0.39	**0.54**‡	**0.53**‡	0.15	0.18
	Base DSPy	0.52	0.43	0.39	**0.44**	0.48	0.52	**0.21**	**0.18**
	Opt. DSPy	0.63	**0.60**‡	**0.41**	0.40	0.46	0.50	0.20	0.16
GPT-5.4-mini	Manual	**0.60**‡	0.48	0.36	0.36	0.45	0.38	0.12	0.08
	Base DSPy	0.53	0.41	0.33	0.36	0.45	0.33	**0.15**	0.11
	Opt. DSPy	0.49	**0.55**‡	**0.49**†	**0.42**‡	**0.52**	**0.48**‡	0.12	**0.14**
Gemini 3.1 Flash-Lite	Manual	**0.64**‡	0.42	0.41	0.42	0.61	0.51	**0.23**	0.09
	Base DSPy	0.51	0.34	0.41	0.44	0.55	0.54	0.16	0.17
	Opt. DSPy	0.56	**0.59**‡	**0.51**‡	**0.53**‡	**0.68**‡	**0.65**‡	0.21	**0.25**†
Claude Sonnet 4.6	Manual	**0.56**	0.43	0.43	0.46	0.61	0.51	**0.11**	0.08
	Base DSPy	0.46	0.37	0.43	0.48	**0.63**	0.52	0.10	**0.12**
	Opt. DSPy	0.53	**0.53**‡	**0.48**	**0.53**‡	0.59	**0.61**‡	0.08	0.10
Qwen3:8B-q4_K_M	Manual	**0.60**‡	**0.54**	**0.32**	0.28	0.44	0.39	**0.19**	0.16
	Base DSPy	0.53	0.41	0.31	0.30	0.38	0.30	0.16	0.13
	Opt. DSPy	0.51	0.48	0.31	**0.31**	**0.55**‡	**0.45**‡	0.15	**0.18**

Table 3: Summary performance for the LQA task, macro-averaged across six target locales. Uni = unified (single-stage); Dec = decomposed (two-stage). N: 1250 (ar=91, de-DE=353, es-ES=362, ru-RU=80, sv-SE=245, zh-CN=119). Best value per model and column in **bold**. † $p < 0.05$, ‡ $p < 0.01$ vs. the next-best configuration (paired stratified bootstrap, 1,000 iterations, Holm–Bonferroni corrected per model).

models, more consistently for Gemini 3.1 Flash-Lite. The degree to which optimization improves over base DSPy signatures also varies by task: gains are largest and most consistent in translation and LQA, while terminology insertion shows smaller improvements. The magnitude of optimization gains also varies by model: Qwen3:8B-q4_K_M, while performing worse than proprietary models on translation and terminology insertion, narrowed the gap on several LQA metrics, with optimization over base DSPy producing the largest single gain observed across all tasks and models, suggesting optimization may yield larger returns over base DSPy when the model is weaker.

Practical implications. We highlight three concrete takeaways for practitioners. First, unified single-stage systems generally match or outperform decomposed multi-stage pipelines—most clearly for error detection in LQA (0.05–0.20 points on Detection F1) and edit distance in terminology insertion—suggesting end-to-end designs should be preferred unless there is a specific reason to decompose. Second, GEPA reliably elevates minimal DSPy signatures across tasks and models, making it a strong default when labeled data is available. Third, within unified systems, expert prompts retain an advantage for LQA error detection while optimization helps characterization —no single approach wins on every sub-metric.

This suggests that combining the two, for example by seeding GEPA with expert prompts and letting it refine instructions, may be especially valuable for LQA. Practitioners should choose between approaches, or combine them, based on their available resources (labeled data, human effort, compute) and task requirements, rather than treating one as inherently superior.

Limitations. This study has several limitations. First, the comparison involves an inherent asymmetry in how each approach uses labeled data. GEPA optimization programmatically exploits gold-standard training and validation splits through automated feedback loops, whereas expert prompt engineering draws on domain knowledge, iterative testing, and, where available, selective inspection of examples. Second, the two approaches are not mutually exclusive: a domain expert could use GEPA optimization as a starting point and manually refine the resulting prompts, or conversely, use expert-crafted prompts as seed inputs for automatic optimization. Our study treats them as independent conditions, but a combined workflow may outperform either in isolation. Third, several design choices were scoped for practical reasons: we evaluate a single optimizer (GEPA) in its "light" mode with the `dspy.Predict` module, we do not evaluate few-shot or chain-of-thought manual prompts, and all evaluation is automatic without human judgments. Each of these choices may underestimate the ceiling of either approach: stronger optimizers, richer expert baselines, and human evaluation could all shift the results, but they were necessary to keep the study tractable across three tasks, five models, and multiple conditions. Fourth, all language pairs are English-centric, reflecting the availability of gold-standard data per task; results may not generalize to non-English-centric directions.

Future work. Future work could address these gaps by comparing additional optimizers and optimization modes, evaluating few-shot expert prompts, evaluating hybrid workflows that combine expert knowledge with automatic optimization, and conducting a cost-benefit analysis of manual versus automatic prompt development across the product lifecycle.

6 Sustainability Statement

This study involves no model training or fine-tuning; all experiments consist of inference-time prompt optimization and evaluation. Four of five model configurations rely on commercial APIs (OpenAI, Google, Anthropic), for which precise energy consumption cannot be estimated. The fifth uses locally hosted open-weight models run via Ollama on consumer-grade laptops. GEPA's "light" optimization mode minimizes candidate programs per run, keeping the overall computational footprint modest. We acknowledge that repeated API calls across multiple tasks, models, and optimization objectives contribute to cumulative energy use that cannot be precisely quantified for API-based configurations.

References

Agrawal, Lakshya A, 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. `https://arxiv.org/abs/2507.19457`.

Deng, Mingkai, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In Goldberg, Yoav, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates, December. Association for Computational Linguistics.

Errica, Federico, Davide Sanvito, Giuseppe Siracusano, and Roberto Bifulco. 2025. What did I do wrong? quantifying LLMs' sensitivity and consistency to prompt engineering. In Chiruzzo, Luis, Alan Ritter, and Lu Wang, editors, *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 1543–1558, Albuquerque, New Mexico, April. Association for Computational Linguistics.

Fernandes, Patrick, Daniel Deutsch, Mara Finkelstein, Parker Riley, André Martins, Graham Neubig, Ankush Garg, Jonathan Clark, Markus Freitag, and Orhan Firat. 2023. The devil is in the errors: Leveraging large language models for fine-grained machine translation evaluation. In Koehn, Philipp, Barry Haddow, Tom Kocmi, and Christof Monz, editors, *Proceedings of the Eighth Conference on Ma-*

chine Translation, pages 1066–1083, Singapore, December. Association for Computational Linguistics.

Freitag, Markus, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9:1460–1474.

Guerreiro, Nuno M., Ricardo Rei, Daan van Stigt, Luisa Coheur, Pierre Colombo, and André F. T. Martins. 2023. xcomet: Transparent machine translation evaluation through fine-grained error detection. https://arxiv.org/abs/2310.10482.

Guo, Qingyan, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2025. Evoprompt: Connecting llms with evolutionary algorithms yields powerful prompt optimizers. https://arxiv.org/abs/2309.08532.

He, Zeyu, Saniya Naphade, and Ting-Hao Kenneth Huang. 2025. Prompting in the dark: Assessing human performance in prompt engineering for data labeling when gold labels are absent. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, page 1–33. ACM, April.

Hendy, Amr, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. https://arxiv.org/abs/2302.09210.

Huang, Xu, Zhirui Zhang, Xiang Geng, Yichao Du, Jiajun Chen, and Shujian Huang. 2024. Lost in the source language: How large language models evaluate the quality of machine translation. In Ku, Lun-Wei, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3546–3562, Bangkok, Thailand, August. Association for Computational Linguistics.

Jiao, Wenxiang, Wenxuan Wang, Jen tse Huang, Xing Wang, Shuming Shi, and Zhaopeng Tu. 2023. Is chatgpt a good translator? yes with gpt-4 as the engine. https://arxiv.org/abs/2301.08745.

Khattab, Omar, 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. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. https://arxiv.org/abs/2310.03714.

Kim, Sejoon, Mingi Sung, Jeonghwan Lee, Hyunkuk Lim, and Jorge Gimenez Perez. 2024. Efficient terminology integration for LLM-based translation in specialized domains. In Haddow, Barry, Tom Kocmi, Philipp Koehn, and Christof Monz, editors, *Proceedings of the Ninth Conference on Machine Translation*, pages 636–642, Miami, Florida, USA, November. Association for Computational Linguistics.

Kocmi, Tom and Christian Federmann. 2023a. GEMBA-MQM: Detecting translation quality error spans with GPT-4. In Koehn, Philipp, Barry Haddow, Tom Kocmi, and Christof Monz, editors, *Proceedings of the Eighth Conference on Machine Translation*, pages 768–775, Singapore, December. Association for Computational Linguistics.

Kocmi, Tom and Christian Federmann. 2023b. Large language models are state-of-the-art evaluators of translation quality. https://arxiv.org/abs/2302.14520.

Kocmi, Tom, Ekaterina Artemova, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Konstantin Dranch, Anton Dvorkovich, Sergey Dukanov, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Marzena Karpinska, Philipp Koehn, Howard Lakougna, Jessica Lundin, Christof Monz, Kenton Murray, Masaaki Nagata, Stefano Perrella, Lorenzo Proietti, Martin Popel, Maja Popović, Parker Riley, Mariya Shmatova, Steinthór Steingrímsson, Lisa Yankovskaya, and Vilém Zouhar. 2025. Findings of the WMT25 general machine translation shared task: Time to stop evaluating on easy test sets. In Haddow, Barry, Tom Kocmi, Philipp Koehn, and Christof Monz, editors, *Proceedings of the Tenth Conference on Machine Translation*, pages 355–413, Suzhou, China, November. Association for Computational Linguistics.

Li, Xiang Lisa and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In Zong, Chengqing, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August. Association for Computational Linguistics.

Liu, Xiao, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In Muresan, Smaranda, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland, May. Association for Computational Linguistics.

Lommel, Arle, Aljoscha Burchardt, and Hans Uszkoreit. 2014. Multidimensional quality metrics (mqm):

A framework for declaring and describing translation quality metrics. *Tradumàtica: tecnologies de la traducció*, 12:455–463.

Lu, Qingyu, Liang Ding, Kanjian Zhang, Jinxia Zhang, and Dacheng Tao. 2025. MQM-APE: Toward high-quality error annotation predictors with automatic post-editing in LLM translation evaluators. In Rambow, Owen, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5570–5587, Abu Dhabi, UAE, January. Association for Computational Linguistics.

Mizrahi, Moran, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? a call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949.

Moslem, Yasmin, Gianfranco Romani, Mahdi Molaei, John D. Kelleher, Rejwanul Haque, and Andy Way. 2023. Domain terminology integration into machine translation: Leveraging large language models. In Koehn, Philipp, Barry Haddow, Tom Kocmi, and Christof Monz, editors, *Proceedings of the Eighth Conference on Machine Translation*, pages 902–911, Singapore, December. Association for Computational Linguistics.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. https://arxiv.org/abs/2207.04672.

Popović, Maja. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *WMT@EMNLP*.

Post, Matt. 2018. A call for clarity in reporting bleu scores. https://arxiv.org/abs/1804.08771.

Pryzant, Reid, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. https://arxiv.org/abs/2305.03495.

Rei, Ricardo, Craig Stewart, Ana C Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In Webber, Bonnie, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online, November. Association for Computational Linguistics.

Sahoo, Pranab, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2025. A systematic survey of prompt engineering in large language models: Techniques and applications. https://arxiv.org/abs/2402.07927.

Sclar, Melanie, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. https://arxiv.org/abs/2310.11324.

Sellam, Thibault, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In Jurafsky, Dan, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online, July. Association for Computational Linguistics.

Shi, Weijia, Xiaochuang Han, Hila Gonen, Ari Holtzman, Yulia Tsvetkov, and Luke Zettlemoyer. 2022. Toward human readable prompt tuning: Kubrick's the shining is a good movie, and a good prompt too? https://arxiv.org/abs/2212.10539.

Shin, Taylor, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In Webber, Bonnie, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online, November. Association for Computational Linguistics.

Voronov, Anton, Lena Wolf, and Max Ryabinin. 2024. Mind your format: Towards consistent evaluation of in-context learning improvements. https://arxiv.org/abs/2401.06766.

White, Jules, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpta prompt pattern catalog to enhance prompt engineering with chatgpt. https://arxiv.org/abs/2302.11382.

Yan, Jianhao, Pingchuan Yan, Yulong Chen, Jing Li, Xianchao Zhu, and Yue Zhang. 2024. Benchmarking gpt-4 against human translators: A comprehensive evaluation across languages, domains, and expertise levels. https://arxiv.org/abs/2411.13775.

Yang, Chengrun, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. https://arxiv.org/abs/2309.03409.

Yuksekgonul, Mert, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic "differentiation" via text. https://arxiv.org/abs/2406.07496.

Zhang, Ningyu, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022a. Differentiable prompt makes pre-trained language models better few-shot learners. https://arxiv.org/abs/2108.13161.

Zhang, Tianjun, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. 2022b. Tempera: Test-time prompting via reinforcement learning. https://arxiv.org/abs/2211.11890.

Zhou, Yulin, Yiren Zhao, Ilia Shumailov, Robert Mullins, and Yarin Gal. 2023. Revisiting automated prompting: Are we actually doing better? https://arxiv.org/abs/2304.03609.

Zhu, Wenhao, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2024. Multilingual machine translation with large language models: Empirical results and analysis. In Duh, Kevin, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2765–2781, Mexico City, Mexico, June. Association for Computational Linguistics.