Evaluating Robustness of Large Language Models Against Multilingual Typographical Errors

Anonymous ACL submission

Abstract

Large language models (LLMs) are increasingly deployed in multilingual, real-world applications with user inputs – naturally introducing *typographical errors* (typos). Yet most benchmarks assume clean input, leaving the robustness of LLMs to typos across languages largely underexplored. To address this gap, we introduce **MULTYPO**, a multilingual typo generation algorithm that simulates human-like errors based on language-specific keyboard layouts and typing behavior. We evaluate 18 open-source LLMs across three model families and five downstream tasks spanning language inference, multi-choice question answering, mathematical reasoning, and machine translation tasks. Our results show that typos consistently degrade performance, particularly in generative tasks and those requiring reasoning – while the natural language inference task is comparatively more robust. Instruction tuning improves clean-input performance but may increase brittleness under noise. We also observe language-dependent robustness: high-resource languages are generally more robust than low-resource ones, and translation from English is more robust than translation into English. Our findings underscore the need for noise-aware training and multilingual robustness evaluation. We will make our code and data publicly available.

1 Introduction

LLMs are increasingly deployed in real-world applications such as chatbots, translation tools, and search engines (Dam et al., 2024; Naveed et al., 2024; Raza et al., 2025), where users input text via keyboards in a wide range of languages. In such settings, *typographical errors* (typos) are a natural part of user input – arising from slips, fast typing, or unfamiliarity with keyboard layouts (Wengelin, 2007; Conijn et al., 2019; Shi et al., 2025). Despite this, most LLM evaluations assume clean, error-free input and report only a single aggregate statistic on a held-out set, thereby overlooking this ubiquitous source of noise (Sun et al., 2020; Moradi and Samwald, 2021; Wang et al., 2024) and often overestimating real-world performance (Ribeiro et al., 2020; Hedderich et al., 2022; Zhao et al., 2024) Robustness to typos is not just a usability concern; it is essential for ensuring reliable model behavior, maintaining user trust, and delivering consistent downstream performance in practical deployments.

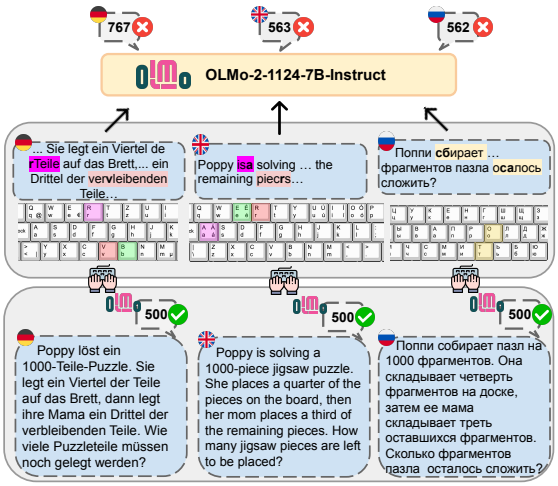

Figure 1: Illustration of the impact of real-world typographical errors. Humans often make typos on language-specific keyboard layouts, and once such errors are introduced, models can fail across languages. In this example, the model cannot generate the correct answer ("500") under typos in English, German, and Russian.

Prior work on robustness evaluation has largely focused on adversarial or synthetic perturbations in an English-centric manner (Gao et al., 2018; Wang et al., 2023; Gan et al., 2024; Zhu et al., 2024; Zhang et al., 2025). As a result, we know little about how robust LLMs are against realistic typos in a multilingual context. Models can generate wrong answers with simple input textual perturbations across languages, as shown in Figure 1. Moreover, these approaches often rely on edit-distance heuristics or character-level noise, with little regard

Error	Example Sentence
None	Colorless green ideas smell furiously.
Replacement	Colorless green ideaa smell furiously.
Insertion	Colorless greenm ideas smell furiously.
Deletion	Coorless green ideas smell furiously.
Transposition	Colorless green ideas smell furioulsy.

Table 1: Examples of four different typographical errors.

for typing behavior (e.g., *10-finger typing convention*) based on language-specific keyboard layouts.

To address these gaps, we first introduce **MULTYPO**, a multilingual typo generation algorithm grounded in empirical modeling of typing behavior. Unlike prior work, MULTYPO simulates realistic typos based on actual language-specific keyboard layouts and constraints, allowing us to generate noise that better reflects real-world user patterns across languages. Crucially, we validate typo realism through human evaluation, ensuring our perturbations reflect actual typing behavior. Relying on MULTYPO, we conduct a comprehensive robustness evaluation of 18 open-source LLMs, spanning three major model families: Gemma, Qwen, and OLMo, using both base models and their instruction-tuned versions, across diverse downstream tasks. Moreover, we assess the model robustness under varying levels of typo corruption, and under both zero- and few-shot prompting.

Our experiments yield several key findings. First, typographical errors consistently degrade model performance, particularly on generative tasks and those requiring reasoning (§5.1). Second, model size does not guarantee robustness: both small and large models exhibit noticeable performance drops under typos (§5.2). Third, instruction tuning improves clean-input performance but may also increase vulnerability under noise (§5.3). Fourth, increasing the number of examples in few-shot settings does not improve the robustness against typos (§6.1). Lastly, robustness varies substantially across languages and scripts: for instance, translation *from* English tends to be more robust than translation *into* English (§6.2).

Our contributions are summarized as follows. **(i)** We propose **MULTYPO**, a multilingual typo generation algorithm that simulates realistic human-like errors grounded in language-specific keyboard layouts and typing patterns, and validate its realism through human evaluation. **(ii)** We conduct a comprehensive robustness evaluation suite spanning 18 open-source LLMs from three families (Gemma, Qwen, OLMo), across five downstream tasks. **(iii)** We evaluate robustness under both zero-shot and few-shot prompting, and under varying levels of typographical corruption, enabling fine-grained analysis of model behavior. **(iv)** We will release our code to facilitate further research on multilingual robustness under typographical errors.

2 Background and Related Work

2.1 Typographical Errors

Typographical errors are among the most common forms of natural noise in user-generated text, typically resulting from accidental keystrokes during typing. Early studies (Gardner, 1992; Lisbach and Meyer, 2013) have identified four core types of typos: **replacement**, **insertion**, **deletion**, and **transposition**. *Replacement* errors occur when the intended key is substituted with another, typically an adjacent key. *Insertion* errors arise from unintentionally pressing an adjacent key alongside the intended one, while *deletion* errors involve accidentally omitting a character. *Transposition* errors, frequently attributed to asynchronous hand movements, swap two adjacent characters, particularly those typed by different hands. Examples are illustrated in Table 1. These four categories have been widely adopted and extended in subsequent work on noise modeling and robustness evaluation (Gao et al., 2018; Pruthi et al., 2019; Zhang et al., 2022; Gan et al., 2024). They also form the foundation of our multilingual typo simulation algorithm MULTYPO (cf. §3),[1] which extends this line of work by incorporating language-specific keyboard layouts.

2.2 Related Work

Input Text Perturbations Recent advances in LLMs have sparked growing interest in evaluating their robustness to noisy or manipulated input. A large body of work focuses on *input corruptions*, which aim to degrade model performance by perturbing the input text at various granularities. These include character-level modifications such as misspellings and typographical errors (Gao et al., 2018; Li et al., 2019; Pruthi et al., 2019; Gan et al., 2024), word-level attacks like synonym substitution or word shuffling (Garg and Ramakrishnan, 2020; Jin et al., 2020; Moradi and Samwald, 2021; Zhou et al., 2024), and sentence-level perturbations

[1] For comparison, we implement a *naive baseline* that applies the same operations without layout constraints, in line with prior approaches. We show that our typos better resemble human errors (cf. §3.2) and that models exhibit different robustness to our typos versus the naive ones (cf. §D).

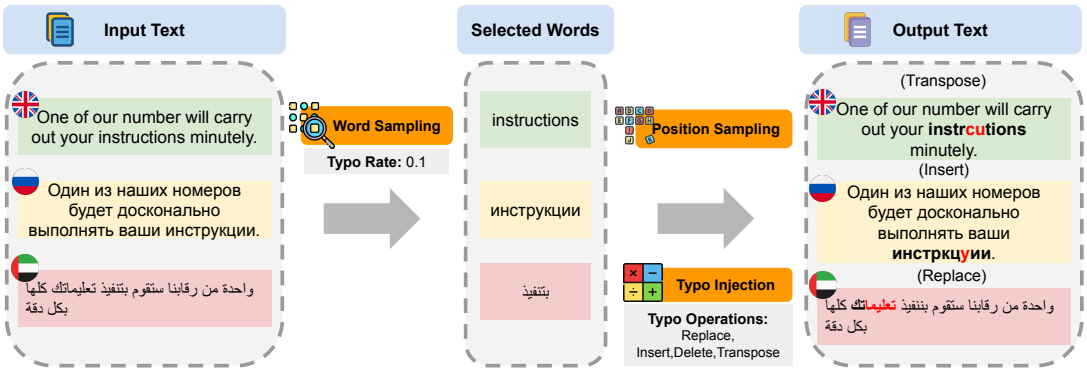

Figure 2: Illustration of the pipeline of MULTYPO: Given an input text and a user-defined typo rate, the algorithm (i) samples words with probability proportional to the square root of the word length, (ii) samples character positions using a position-aware distribution, and (iii) samples one of four typo operations: *replace*, *insert*, *delete*, or *transpose*. Then the algorithm produces a noised text that simulates human-like errors.

through paraphrasing or irrelevant context insertions (Shi et al., 2023a; Lanham et al., 2023; Xu et al., 2024). Even minor punctuation noise can affect LLMs' performance (Abedin et al., 2025). However, most of this research is conducted in English-centric settings, leaving it unclear how typos influence LLMs' robustness across different languages – a gap our work directly addresses.

Multilingual Robustness Evaluation Another line of work has examined the robustness of multilingual models (Cooper Stickland et al., 2023; Aliakbarzadeh et al., 2025). For example, Cooper Stickland et al. (2023) investigated how real-world noise influences encoder-only models such as XLM-R (Conneau et al., 2020) and mBERT (Devlin et al., 2019), and proposed data augmentation with a contrastive loss for pretraining more robust multilingual models. More recently, Aliakbarzadeh et al. (2025) extended this line of investigation to larger multilingual models, demonstrating that performance deteriorates when inputs are corrupted by real-world noise in language understanding tasks. In contrast to them, we explore typographical errors directly by introducing a multilingual, keyboard-aware typo generation algorithm that enables *realistic* and *extensible* simulation across diverse languages. Additionally, our systematic evaluation covers a broader range of tasks beyond language understanding.

3 MULTYPO

This section introduces MULTYPO, a multilingual typo generation algorithm designed to simulate realistic, human-like typos based on language-specific keyboard layouts. Given a clean input text, MULTYPO injects character-level perturbations that mimic natural typing mistakes, producing corrupted outputs that maintain the overall coherence of the original text. We describe the algorithmic design in §3.1, followed by a human evaluation in §3.2 that evaluates how well the generated typos reflect real human typing behavior.

3.1 Algorithm Design

To reflect the real typing behavior of users of different languages, we leverage a keyboard layout database.[2] When inserting typos, special symbols (e.g., punctuation marks or modifier keys such as *Enter*) are excluded. For determining which hand types a specific key, we rely on the standard *10-finger typing convention* for QWERTY English keyboards, according to which characters such as "5TGB" are assigned to the left hand and "6YHN" to the right hand (Logan et al., 2016). For other languages, we adopt the same keyboard-relative hand separation (i.e., left vs. right half, based on key positions) – a common implicit assumption in multilingual layout design – though we validate its appropriateness empirically in our human evaluation. The overall pipeline of MULTYPO is illustrated in Figure 2. Below, we describe the key components in detail, and the description of MULTYPO.

Typo Types We consider four types of typos based on existing literature, as introduced in §2.1: *replacement*, *insertion*, *deletion*, and *transposition*.

- ***Replacement***: A single character in a word is replaced by a neighboring key based on the language-specific keyboard layout.
- ***Insertion***: A randomly selected additional character is inserted immediately after a cor-

[2] https://kbdlayout.info/

rectly typed character, simulating accidental simultaneous keystrokes.

- ***Deletion***: A single character in a word is randomly deleted from the word, simulating the common case where a keypress is missed.

- ***Transposition***: Two adjacent characters are swapped. We constrain this to occur only between characters typed with *different* hands, based on the 10-finger typing convention.

Ignoring String Sets To avoid corrupting tokens that are critical for downstream understanding, especially numbers, we define a language-specific set of strings to ignore during typo insertion.[3] These sets include numerical expressions commonly used across languages – both in digit form (e.g., 1, 2, 3) and in word form (e.g., "three", "hundred", "million") (see Figure 8 in §A). During typo generation, any word that matches or contains a string in the set is excluded from being inserted with typos.

Length-Aware Sampling Probability Rather than treating all words equally, we assign each word a sampling probability proportional to the square root of its length: $\frac{\sqrt{|w|}}{\sum_w \sqrt{|w|}}$ (normalized over all words in a given text), reflecting the tendency for longer words to attract more typos (Peterson, 1986; Kukich, 1992). In addition, when selecting a specific character position within a word to insert or modify, we also consider position-dependent weights. Following observations from Lisbach and Meyer (2013), which show that errors are more likely to occur toward the middle or end of a word, we assign a non-uniform probability distribution over character indices, with details provided in §A.

Algorithm Description Given an input sequence $S = \{w_1, w_2, \ldots, w_n\}$ of n words, our algorithm begins by computing the number of typos to insert, determined by a user-defined corruption ratio $\tau \in [0, 1]$ and the total number of words n, rounded to the nearest integer. Each word w_i is assigned a sampling probability proportional to the square root of its character length $\sqrt{|w|}$, as described earlier, and candidates are sampled accordingly. For each selected word, we sample one of four typo operations: *replace*, *insert*, *delete*, or *transpose*. The specific character position within the word is sampled based on a length-aware, position-dependent distribution that favors later positions. Once the position in a word is determined, the algorithm applies the selected typo operation to that position. After each successful typo insertion, the corresponding word's sampling weight is halved to encourage distributional diversity. The algorithm proceeds iteratively until either the target number of typos is reached or a maximum retry threshold is exceeded. The pipeline of MULTYPO is illustrated in Figure 2.

[3] While this makes the noise slightly less realistic, it ensures that benchmark results reflect robustness to typos rather than being skewed by altered numeric values in the prompt.

Language	Multypo (avg.)	Naive (avg.)	Significance
German	8.93	5.27	***
English	10.00	4.79	***
Russian	9.67	7.67	**
Hindi	9.40	6.67	**
French	9.60	6.67	***
Greek	8.07	5.93	*
Arabic	6.00	6.60	

Table 2: Average number of sentences judged as "natural" out of 15 corrupted sentences per system, with significance from paired t-tests. Stars denote the significance levels: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

3.2 Human Evaluation on Typo Naturalness

To further assess the realism of MULTYPO-generated typos, we conduct a human evaluation comparing MULTYPO against a *naive baseline* that applies the same four operations (*insertion*, *deletion*, *substitution*, *transposition*) but without considering keyboard layout constraints. For each language, we sample 30 sentences from Flores200 (NLLB Team et al., 2022), split into two equal halves: 15 sentences corrupted by MULTYPO and 15 by the naive baseline. Within each set, we balanced the number of sentences across 3 corruption levels (0.1, 0.4, and 0.7; five sentences per level), while ensuring that sentence length distributions remained comparable across the two conditions. At least 15 participants in each language were asked to judge whether the typos in each sentence appeared *natural* or *unnatural*. This binary judgment provides a direct measure of how human-like the errors appear. We collected annotations across seven languages: Arabic, German, Greek, English, French, Hindi, and Russian (details provided in §B).

Table 2 summarizes the results across languages. In six of the seven cases, MULTYPO is judged significantly more natural than the random baseline in the paired t-test (at least $p < 0.05$). Arabic is the only exception, where ratings slightly favored the baseline, but without statistical significance. We include it in our further experiments for completeness, though results might need to be interpreted

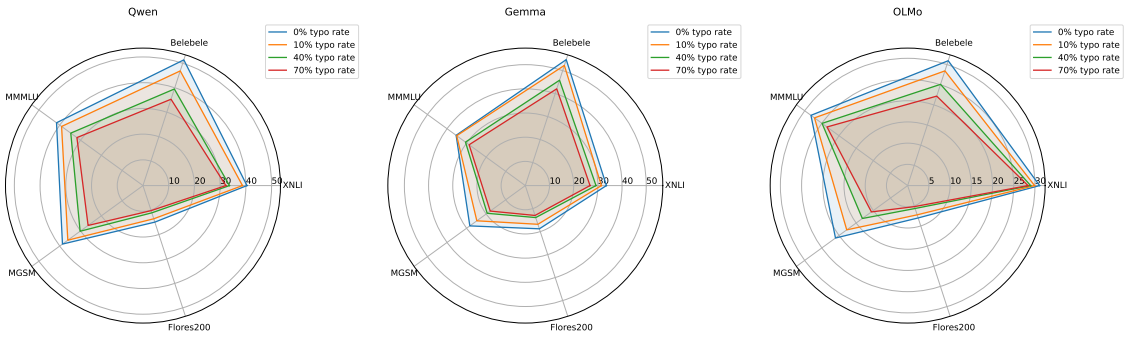

Figure 3: Performance under different typo rates (0, 0.1, 0.4, and, 0.7) averaged across languages for each task.

with caution. Taken together, this human evaluation confirms that MULTYPO can generally generate typos perceived as more human-like across languages than a naive baseline process that does not consider keyboard layout constraints. In §D, we also show that models exhibit different robustness to our typos versus the naive ones: models are more robust to typos generated by MULTYPO, possibly due to the exposure of similar typos – real-world human typos – in the pretraining phase.

4 Experimental Setup

This section outlines our evaluation setup, where we apply MULTYPO to inject human-like typos into diverse downstream tasks and assess the robustness of different LLMs to these perturbations.

4.1 Languages

We consider 12 languages spanning 7 language families and written in 7 different scripts, with a focus on alphabet-based writing systems where typos are primarily influenced by keyboard layout. The set of supported languages by MULTYPO includes Arabic (**ara_Arab**), Armenian (**hye_Armn**), Bengali (**ben_Beng**), English (**eng_Latn**), French (**fra_Latn**), Georgian (**kat_Geor**), German (**deu_Latn**), Greek (**ell_Grek**), Hebrew (**heb_Hebr**), Hindi (**hin_Deva**), Russian (**rus_Cyrl**), and Tamil (**tam_Taml**).

4.2 Models

We evaluate 18 decoder-only language models from 3 model families: **Gemma** (Gemma Team et al., 2025), **Qwen** (Yang et al., 2025), and **OLMo** (Team OLMo et al., 2025). Models from the first two families are pretrained on highly multilingual corpora, while OLMo is pretrained on English-centric data. For the Gemma family, we consider `gemma-3-1b-pt`, `gemma-3-4b-pt`, and `gemma-3-12b-pt`. For the Qwen family, we consider `Qwen3-1.7B-Base`, `Qwen3-4B-Base`, and `Qwen3-8B-Base`. Finally for the OLMo family, we consider `OLMo-2-0425-1B`, `OLMo-2-1124-7B`, and `OLMo-2-1124-13B`. For each model above, we also consider its corresponding instruction-tuned version, aiming to systematically investigate the robustness against multilingual typos of models across size, family, and training strategies.

4.3 Dataset

To evaluate robustness under multilingual typos, we use six datasets spanning four task types: natural language inference (**XNLI** (Conneau et al., 2018)), multiple-choice questions answering (**Belebele** (Bandarkar et al., 2024) and **MMMLU** (Hendrycks et al., 2021)), mathematical reasoning (**MGSM** (Shi et al., 2023b), along with Arabic and Hindi adaptations of GSM8K (Cobbe et al., 2021; Gumma et al., 2024; Omartificial-Intelligence-Space, 2025)), and machine translation (**FLORES200** (NLLB Team et al., 2022)). These datasets are selected to cover diverse tasks and our target languages. Note that the typos are only injected into the dataset instances, but not into other components of the prompt, such as task instructions, to ensure that we are evaluating robustness to input corruption rather than altering the task specification itself. Further details, including language coverage of each dataset and used prompt templates, are provided in §C.

5 Results and Discussion

In this section, we present the results of injecting typos into different datasets. By default, we use 3-shot prompting to ensure a reasonable performance (we further analyze the effect of example-count on robustness in §6.1). In the following parts, we aim to investigate three research questions: **(1)** *Does*

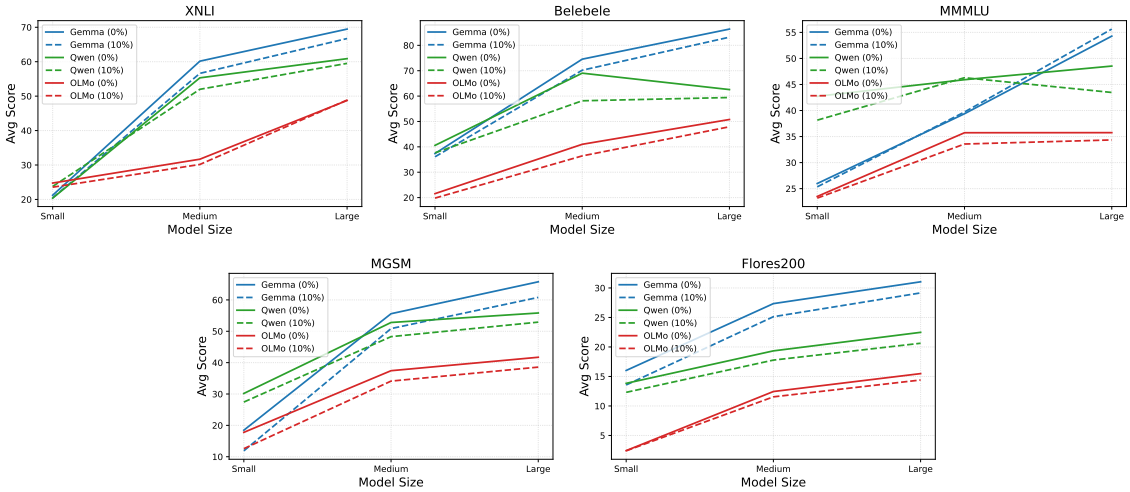

Figure 4: Impact of model size (Small, Medium, Large) on multilingual robustness across five tasks. A different color represents each model family, and two lines are plotted per family: performance on clean input (0%) and input with a 10% typo rate. Larger models generally perform better but also exhibit performance drops under noise.

performance degrade when typographical errors are introduced, and if so, how much? (§5.1); **(2)** *Do larger models present better robustness against typographical errors compared to smaller ones?* (§5.2); and **(3)** *Does instruction-tuning improve the robustness of the models?* (§5.3).

5.1 Performance Drop under Typos

Figure 3 presents the average performance of three model families – Gemma, Qwen, and OLMo – across five multilingual tasks, under varying levels of typographical corruption (0%, 10%, 40%, 70%). Results are aggregated over instruction-tuned models and all supported languages within each task.

Typos consistently degrade performance across all models. Across all families and tasks, even minor typographical noise largely impairs model performance. For example, Qwen achieves over 50 on Belebele in the clean setting, but drops to around 45 with just a 10% typo rate. As noise increases, performance declines continuously. This pattern holds across families and tasks, underscoring a general vulnerability to surface-level perturbations. These findings echo prior monolingual results (Moradi and Samwald, 2021; Wang et al., 2025), and extend them to a multilingual setting.

Robustness varies substantially by task. Typo sensitivity is not uniform across tasks. For instance, XNLI exhibits good robustness: Qwen's performance remains nearly unchanged under 10% noise. Even the OLMo models – despite being primarily monolingual – sustain less than a 10-point absolute drop at the highest noise level. In contrast, tasks involving generative reasoning (e.g., MGSM) are highly susceptible. Qwen's accuracy on MGSM plummets from around 40 (clean) to around 27 (70% noise), suggesting that token-level corruption disrupts multi-step reasoning more than classification-based understanding. These results support earlier, monolingual claims that noisy inputs affect complex reasoning (Gan et al., 2024).

Family	Small	Medium	Large
Gemma	21.46 (-9.9%)	48.50 (-5.7%)	59.11 (-3.7%)
OLMo	16.30 (-9.5%)	29.16 (-7.9%)	36.82 (-4.3%)
Qwen	27.86 (-5.7%)	44.50 (-8.2%)	47.19 (-5.7%)

Table 3: Each cell reports the average score of a model family under a 10% typo rate, with the relative performance drop from clean input shown in parentheses.

Takeaway. While LLMs can often infer intended meaning from noisy input in simpler classification tasks (e.g., natural language inference), reasoning tasks amplify the fragility introduced by typos.

5.2 Model Size Impact on Robustness

Figure 4 presents the average performance when fed with clean inputs and with a small typo rate (10%). We group the models in each model family into different scales (Small, Medium, and Large). Results are averaged across tasks and supported languages, focusing on instruction-tuned models.

Larger models consistently outperform smaller ones, with mild gains in robustness. While larger models consistently outperform smaller ones, also under typo noise, models of all sizes suffer from input perturbations, as shown in Figure 4. To further analyze the effect of model scale on robustness, we compute the relative

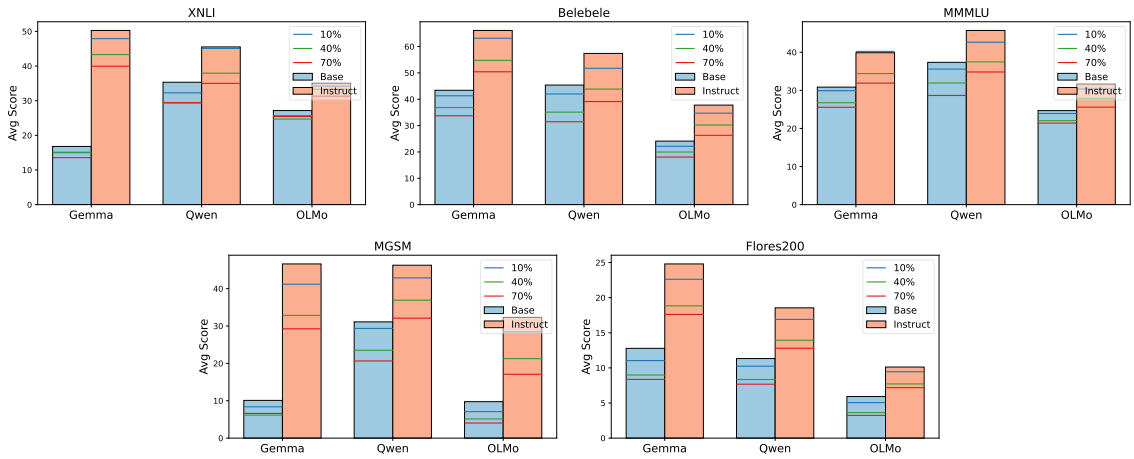

Figure 5: Impact of Instruction-tuning on multilingual robustness. Instruction-tuned models improve the performance, but do not seem to improve the robustness against typos, especially with higher typo rates.

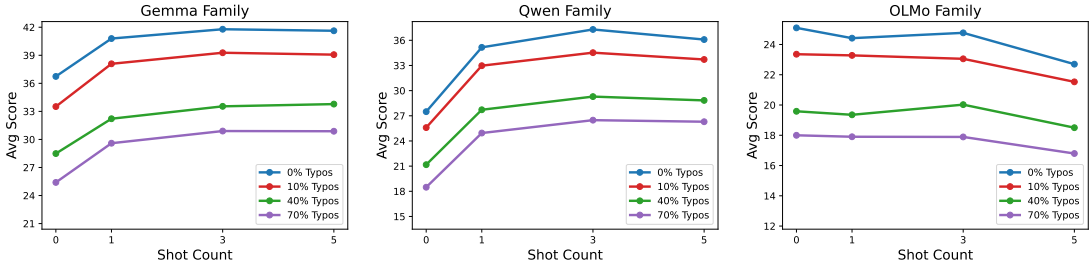

Figure 6: Performance of different model families under different numbers of shots. Increasing the demonstrations slightly improves the performance but does not improve the robustness against typos.

degradation under a 10% typo rate ($\frac{\text{Perf}_{10\%} - \text{Perf}_{0\%}}{\text{Perf}_{0\%}}$). As reported in Table 3, larger models show smaller drops – particularly within the Gemma and OLMo families. E.g., Gemma's relative drop decreases from 9.9% (Small) to 3.7% (Large), suggesting that greater model capacity enables better robustness.

Takeaways. Scaling the model improves task performance under typo noise, but larger models are not immune to noise as well. However, larger models present improved robustness under typos.

5.3 Base vs. Instruction-Tuned Models

Figure 5 presents the performance of pretrained based models and their instruction-tuned versions under varying levels of typographical corruption (0%, 10%, 40%, 70%). Results are averaged over all supported languages within each task.

Instruction-tuned models outperform base models but remain brittle under typos. Instruction-tuning improves overall performance, aligning with prior work (Liu et al., 2023; Chung et al., 2024), which shows that instruction-tuning enhances task-specific multi-step reasoning. Despite clear performance benefits under clean input, instruction-tuned models remain vulnerable to typos. In many cases, the absolute degradation under 10% or 40% noise is as severe as or even worse than their base counterparts. For instance, on MGSM, Gemma's instruction-tuned models drop from around 48 to 33 under 40% corruption. Similar degradation is seen across other families and tasks. This suggests that while instruction-tuned models are better at following complex prompts, they remain equally brittle under surface-level input corruption.

Takeaways. Instruction-tuning boosts performance but does not improve robustness. Current tuning methods prioritize clean prompts and may underprepare models for noisy real-world input.

6 Complementary Analysis

6.1 Does Example Counts Affect Robustness?

Few-shot prompting is known to enhance model performance by providing clearer task formulations and patterns (Brown et al., 2020; Schick and Schütze, 2022). This naturally raises the question: *Can increasing the number of examples also improve robustness against typos?* To explore this, we vary the number of examples in the prompt – 0, 1, 3, and 5 – and evaluate instruction-tuned models across different typo rates, tasks, and languages.

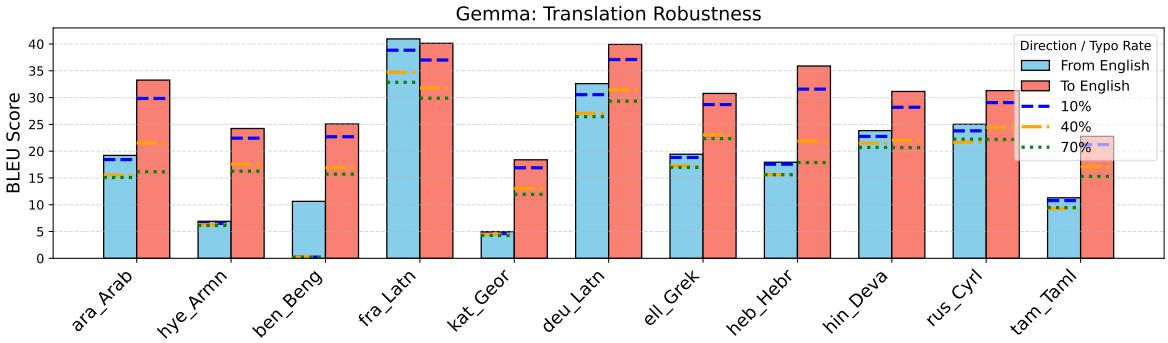

Figure 7: Robustness of **Gemma** models on **Flores200** under different levels of typographical noise. Translation from English seems to be more robust compared to translation to English.

Family	ara_Arab	ben_Beng	deu_Latn	ell_Grek	eng_Latn	fra_Latn	hin_Deva	rus_Cyrl
Gemma	51.3	47.3	53.0	55.3	56.4	53.6	45.6	57.3
	46.3	42.3	52.2	55.4	57.5	52.1	41.3	50.5
	9.7%	10.6%	1.5%	-0.2%	-2.0%	2.8%	9.4%	11.9%
Qwen	47.2	38.3	52.9	47.6	62.3	54.8	41.0	55.0
	44.6	31.6	50.8	44.5	58.8	53.0	36.9	52.4
	5.5%	17.5%	4.0%	6.5%	5.6%	3.3%	10.0%	4.7%
OLMo	30.6	20.3	40.9	39.1	57.5	46.3	26.9	41.9
	28.5	19.1	38.3	37.1	55.8	43.4	24.6	35.3
	6.9%	5.9%	6.4%	5.1%	3.0%	6.3%	8.6%	15.8%

Table 4: Performance when fed with clean input (top row) and with a 10% typo rate (bottom row), aggregated across all tasks and datasets, by language. Only languages supported by at least 3 datasets are considered.

Figure 6 presents the aggregated results.

Across the multilingual models (Gemma and Qwen), increasing the number of examples in few-shot settings leads to consistent performance gains until 3 shots. However, the robustness gap, i.e., the performance drop from clean to noisy inputs, remains nearly unchanged regardless of shot count. For the OLMo family, adding more examples does not seem to help and occasionally harms performance, likely due to its limited multilingual coverage and confusion introduced by prompts in languages that OLMo does not support well. **These findings suggest that while demonstrations improve overall performance, they do not inherently enhance robustness against typos.**

6.2 Which Language is More Sensitive?

We hypothesize that *model robustness to typos is not uniform across languages*, particularly due to data availability. To investigate this, we analyze performance degradation across eight languages that are each supported by at least three datasets. Table 4 presents results aggregated over all tasks. Additionally, we analyze **Flores200** separately to examine how translation direction interacts with typo robustness. Specifically, we compare the performance of Gemma models when translating *from English* vs. *into English*, as shown in Figure 7.

Across all three model families, English consistently exhibits the highest robustness – its relative drop is among the lowest. Other languages that use the Latin script, such as German and French, also show relatively small degradations. In contrast, languages with underrepresented scripts, including Arabic, Hindi, and Bengali, tend to exhibit larger drops. Interestingly, even Russian, despite being high-resource, suffers from sharp degradation (e.g., 11.9% for Gemma). This suggests that **models are more robust in languages with both high data availability and orthographic familiarity (e.g., Latin script)**. Furthermore, Figure 7 shows that translations *from* English are more robust than those *into* English, reinforcing the idea that **typos in lower-resourced or structurally different input languages more severely impair both crosslingual understanding and generation**.

7 Conclusion

We present a comprehensive study on the multilingual robustness of LLMs under simulated, realistic typographical errors. To this end, we introduce MULTYPO, a multilingual typo generation algorithm grounded in language-specific keyboard layouts and human typing behavior. Through extensive evaluation of 18 models across five tasks and three model families, we find that even modest levels of noise can impair model performance – particularly in reasoning-heavy tasks. While larger models and instruction tuning improve performance on clean inputs, they do not necessarily confer greater robustness. Moreover, we identify language-specific sensitivity: models are more resilient in high-resource, Latin-script languages. These findings highlight critical blind spots in current LLM evaluation and motivate future work on noise-aware multilingual pretraining, evaluation, and human-centric error modeling.

Limitations

While our work provides a first step toward multilingual robustness evaluation under human-like typographical errors, we acknowledge that several limitations remain.

First, MULTYPO currently supports a diverse but limited set of typologically diverse languages. To incorporate a new language, one needs to manually specify the corresponding keyboard layout and typing conventions.

Second, our algorithm does not yet support logographic or syllabic writing systems, such as Chinese. This limitation stems from the fundamental differences in input methods – e.g., Chinese characters are typically typed via phonetic systems like *Pinyin* rather than direct keypresses. Modeling such input pipelines requires a fundamentally different corruption strategy. Future work could extend MULTYPO to accommodate these languages by simulating common typing errors in the intermediate input stages (e.g., Pinyin mistyping or candidate misselection).

Third, our human evaluation provides important validation of the realism of MULTYPO, covering multiple languages. However, results for Arabic did not show significant improvements over the naive baseline, suggesting that our typo simulation algorithm MULTYPO may not capture all language-specific properties equally well. However, this does not overshadow the findings that LLMs are not robust to multilingual textual perturbations.

Finally, we focus exclusively on physical keyboards (e.g., QWERTY), while ignoring other input modalities such as touchscreen keyboards on mobile devices. Typing behaviors, error distributions, and auto-correct interference vary substantially across modalities (Jokinen et al., 2021; Shi et al., 2025). Evaluating robustness under such device-dependent noise would further enrich our understanding of LLM performance in real-world settings, which we leave for future work.

Ethical Considerations

Data Annotation Before conducting the human evaluation, all participants were clearly informed about the purpose, procedure, and voluntary nature of the study, and provided their informed consent. For most languages, annotators were recruited via Prolific and compensated fairly at a rate equivalent to approximately £6 per hour (about £1 per completed annotation set) (details are provided in §B). A small portion of participants (around 10%) were personal contacts who volunteered without compensation. No personally identifiable information was collected, and all demographic data (e.g., age, gender) was provided optionally.

Use of AI Assistants The authors acknowledge the use of ChatGPT exclusively for grammar correction, improving the clarity and coherence of the draft, and assisting with code implementation.[4]

References

Zain Ul Abedin, Shahzeb Qamar, Lucie Flek, and Akbar Karimi. 2025. Arithmattack: Evaluating robustness of llms to noisy context in math problem solving. *Preprint*, arXiv:2501.08203.

Amirhossein Aliakbarzadeh, Lucie Flek, and Akbar Karimi. 2025. Exploring robustness of multilingual llms on real-world noisy data. *Preprint*, arXiv:2501.08322.

Yukino Baba and Hisami Suzuki. 2012. How are spelling errors generated and corrected? a study of corrected and uncorrected spelling errors using keystroke logs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 373–377, Jeju Island, Korea. Association for Computational Linguistics.

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)*, pages 749–775, Bangkok, Thailand. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex

[4] https://chatgpt.com/

Castro-Ros, Marie Pellat, Kevin Robinson, and 16 others. 2024. Scaling instruction-finetuned language models. *J. Mach. Learn. Res.*, 25:70:1–70:53.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *CoRR*, abs/2110.14168.

Rianne Conijn, Menno Van Zaanen, Mariëlle Leijten, and Luuk Van Waes. 2019. How to typo? building a process-based model of typographic error revisions. *Journal of Writing Analytics*, 3:69–95.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Asa Cooper Stickland, Sailik Sengupta, Jason Krone, Saab Mansour, and He He. 2023. Robustification of multilingual language models to real-world noise in crosslingual zero-shot settings with robust contrastive pretraining. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1375–1391, Dubrovnik, Croatia. Association for Computational Linguistics.

Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. 2024. A complete survey on llm-based ai chatbots. *Preprint*, arXiv:2406.16937.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Esther Gan, Yiran Zhao, Liying Cheng, Mao Yancan, Anirudh Goyal, Kenji Kawaguchi, Min-Yen Kan, and Michael Shieh. 2024. Reasoning robustness of LLMs to adversarial typographical errors. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10449–10459, Miami, Florida, USA. Association for Computational Linguistics.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-Box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers. In *2018 IEEE Security and Privacy Workshops, SP Workshops 2018, San Francisco, CA, USA, May 24, 2018*, pages 50–56. IEEE Computer Society.

Sylvia A. Gardner. 1992. Spelling Errors in Online Databases: What the Technical Communicator Should Know. *Technical Communication*, 39(1):50–53.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 25 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Varun Gumma, Pranjal A. Chitale, and Kalika Bali. 2024. Towards Inducing Document-Level Abilities in Standard Multilingual Neural Machine Translation Models. *Preprint*, arXiv:2408.11382.

Michael A. Hedderich, Jonas Fischer, Dietrich Klakow, and Jilles Vreeken. 2022. Label-descriptive patterns and their application to characterizing classification errors. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 8691–8707. PMLR.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8018–8025. AAAI Press.

Jussi Jokinen, Aditya Acharya, Mohammad Uzair, Xinhui Jiang, and Antti Oulasvirta. 2021. Touchscreen typing as optimal supervisory control. In *CHI '21: CHI Conference on Human Factors in Computing Systems, Virtual Event / Yokohama, Japan, May 8-13, 2021*, pages 720:1–720:14. ACM.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *ACM computing surveys (CSUR)*, 24(4):377–439.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *Preprint*, arXiv:2307.13702.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society.

Bertrand Lisbach and Victoria Meyer. 2013. *Linguistic Identity Matching*. Springer.

Hanmeng Liu, Zhiyang Teng, Leyang Cui, Chaoli Zhang, Qiji Zhou, and Yue Zhang. 2023. LogiCoT: Logical chain-of-thought instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2908–2921, Singapore. Association for Computational Linguistics.

Gordon D Logan, Jana E Ulrich, and Dakota RB Lindsey. 2016. Different (key) strokes for different folks: How standard and nonstandard typists balance fitts' law and hick's law. *Journal of Experimental Psychology: Human Perception and Performance*, 42(12):2084.

Peter F MacNeilage. 1964. Typing errors as clues to serial ordering mechanisms in language behaviour. *Language and speech*, 7(3):144–159.

Milad Moradi and Matthias Samwald. 2021. Evaluating the robustness of neural language models to input perturbations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1558–1570, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2024. A comprehensive overview of large language models. *Preprint*, arXiv:2307.06435.

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, and 20 others. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Omartificial-Intelligence-Space. 2025. Arabic GSM8K: Arabic Grade School Math Dataset. `https://huggingface.co/datasets/Omartificial-Intelligence-Space/Arabic-gsm8k`.

James L Peterson. 1986. A note on undetected typing errors. *Communications of the ACM*, 29(7):633–637.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

Mubashar Raza, Zarmina Jahangir, Muhammad Bilal Riaz, Muhammad Jasim Saeed, and Muhammad Awais Sattar. 2025. Industrial applications of large language models. *Scientific Reports*, 15(1):13755.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2022. True few-shot learning with Prompts—A real-world perspective. *Transactions of the Association for Computational Linguistics*, 10:716–731.

Danqing Shi, Yujun Zhu, Francisco Erivaldo Fernandes Junior, Shumin Zhai, and Antti Oulasvirta. 2025. Simulating errors in touchscreen typing. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, CHI 2025, YokohamaJapan, 26 April 2025- 1 May 2025*, pages 1086:1–1086:13. ACM.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023a. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023b. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020. Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert. *Preprint*, arXiv:2003.04985.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, and 21 others. 2025. 2 olmo 2 furious. *Preprint*, arXiv:2501.00656.

Bin Wang, Chengwei Wei, Zhengyuan Liu, Geyu Lin, and Nancy F. Chen. 2024. Resilience of large language models for noisy instructions. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11939–11950, Miami, Florida, USA. Association for Computational Linguistics.

Haoyu Wang, Guozheng Ma, Cong Yu, Ning Gui, Linrui Zhang, Zhiqi Huang, Suwei Ma, Yongzhe Chang, Sen Zhang, Li Shen, Xueqian Wang, Peilin Zhao, and Dacheng Tao. 2025. Are large language models really robust to word-level perturbations? *Trans. Mach. Learn. Res.*, 2025.

Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, Binxin Jiao, Yue Zhang, and Xing Xie. 2023. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. *Preprint*, arXiv:2302.12095.

Åsa Wengelin. 2007. The word-level focus in text production by adults with reading and writing difficulties. In *Writing and cognition*, pages 67–82. Brill.

Xilie Xu, Keyi Kong, Ning Liu, Lizhen Cui, Di Wang, Jingfeng Zhang, and Mohan S. Kankanhalli. 2024. An LLM can fool itself: A prompt-based adversarial attack. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Kun Zhang, Le Wu, Kui Yu, Guangyi Lv, and Dacao Zhang. 2025. Evaluating and improving robustness in large language models: A survey and future directions. *Preprint*, arXiv:2506.11111.

Yunxiang Zhang, Liangming Pan, Samson Tan, and Min-Yen Kan. 2022. Interpreting the robustness of neural NLP models to textual perturbations. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3993–4007, Dublin, Ireland. Association for Computational Linguistics.

Raoyuan Zhao, Abdullatif Köksal, Yihong Liu, Leonie Weissweiler, Anna Korhonen, and Hinrich Schuetze. 2024. SynthEval: Hybrid behavioral testing of NLP models with synthetic CheckLists. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7017–7034, Miami, Florida, USA. Association for Computational Linguistics.

Zihao Zhou, Qiufeng Wang, Mingyu Jin, Jie Yao, Jianan Ye, Wei Liu, Wei Wang, Xiaowei Huang, and Kaizhu Huang. 2024. Mathattack: Attacking large language models towards math solving ability. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 19750–19758. AAAI Press.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Gong, and Xing Xie. 2024. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. In *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis, LAMPS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, pages 57–68. ACM.

A Additional Details of MULTYPO

Neighbors of Keys In our work, we define a character's neighbors (which are used in typo operation *replace* and *insert*) as the keys that are immediately adjacent to it on the same row of the keyboard – specifically, the key to its left and the key to its right, based on the empirical findings that horizontal neighbor errors are more common (MacNeilage, 1964).[5] Keys corresponding to non-alphabetic symbols or function keys (such as *Enter* or *Tab*) are not considered, and characters that are absent from the specified keyboard layout simply have no neighbors. This definition provides a consistent notion of adjacency across languages and platforms, avoiding complications introduced by layout-specific variations in the placement of special characters (e.g., brackets on Windows vs. macOS).

Ignoring String Sets As described in §3.1, we exclude several words from being selected as the candidates to insert typos. These words are mainly commonly used numerical expressions across languages. We exclude them since they are typically critical for downstream understanding. The set for each language is visualized in Figure 8.

[5]To reflect the real typing behavior of users of different languages, we leverage a keyboard layout database from `https://kbdlayout.info/`, which uses data sourced from Windows version 10.0.27729.1000

Figure 8: Ignoring String Sets across all languages that are considered in MULTYPO.

Sampling In MULTYPO, there are three places where we introduce randomness by sampling: *word sampling*, *position sampling*, and *typo operation sampling*. We introduce the details of each sampling process in the following.

- ***Word Sampling*:** We assign each word a sampling probability proportional to the square root of its length: $\sqrt{|w|}$. We then normalize the probability by dividing the sum over all words: $\sum_w \sqrt{|w|}$. This probability reflects the empirically observed tendency for longer words to attract more typos.

- ***Position Sampling*:** When selecting a specific character position within a word to insert or modify a character, we consider position-dependent weights: the first character is assigned a weight of 0 and is never selected; the second character receives a weight of 0.1; the final character receives 0.2; and all intermediate positions are linearly interpolated between these values. This empirical setup is based on findings from Lisbach and Meyer (2013) that word-initial errors are rare. The probability of each position is then its normalized weight.

- ***Typo Operation Sampling*:** *Insertion* operations are sampled with a probability of 15.25%, while *replacement*, *deletion*, and *transposition* are each assigned a probability of 28.25%. This skewed distribution reflects empirical observations that insertion errors are less common than the other three error types according to the findings from Baba and Suzuki (2012).

Validating Operation In our implementation, instead of sampling all candidate words at once, we iteratively select a word at a time and insert a typo into it, guaranteeing an adequate number of typos according to the user specification (typo rate). Therefore, when inserting a typo into a word, we perform a validity check before applying each operation. Because errors are introduced iteratively, unconstrained edits could yield implausible outcomes – for example, replacing the initial *w* in *word* with *e* to obtain *eord*, and then replacing *e* back with *w*, which would be counted as two errors despite leaving the word unchanged. The validity check prevents such contradictions and filters out unlikely multi-step substitutions, ensuring that the final set of typos is consistent with natural typing patterns.

Special Cases and Strategies We handle several edge cases to keep the typographical errors meaningful and realistic:

- If the selected word is only *one character long*, or if it contains an item of the predefined Ignoring String Sets, another word will be selected.

- If the *transpose* operation is selected and the word is not the final word in the sentence, a whitespace character is appended to the end of the word. This is done to facilitate the detection of cross-hand key pairs, as previously described, unless the whitespace has already been appended in an earlier iteration.

- If the typo operation is deemed invalid or if the modified word is identical to the original (indicating no actual change occurred), another typo function is selected for the same word.

B Details of Human Evaluation

Overview We designed a lightweight web interface to collect judgments of typo naturalness. Participants first selected their evaluation language (**English**, **German**, **French**, **Greek**, **Russian**, **Arabic**, **Hindi**), filled in basic demographic information (*age*, *gender*, *nationality*, *fluency* in the selected language),[6] , as shown in Figure 9 and then completed 30 annotation trials. Each trial displayed a single corrupted sentence from Flores200 (NLLB Team et al., 2022), and participants judged whether the typos appeared *natural* or *unnatural*, as shown in Figure 10. At the end of the evaluation, the participant needed to provide a confidence rating (1–5) (the higher, the more confident).

Task Setup. In total, 30 sentences were sampled from Flores200, with 15 corrupted by MULTYPO and 15 by the random baseline (5 sentences at 10%, 40%, and 70% typo rates for each system). Sentences were balanced for length across the two conditions. The order of the sentences is randomized for each participant to avoid a systematic learning effect while going through the sentences. Participants completed the annotation in ∼5-8 minutes. Participants were recruited through personal contacts (∼10%, mainly for English and German), extended through recruitment on Prolific.[7] We compensated crowdworkers at a rate equivalent to about £6 per hour, which corresponds to roughly £1 per annotation set (30 sentences).

Participants. We collected at least 15 valid responses per language.[8] Table 5 summarizes the fluency levels and self-reported confidence across languages. Table 6 and Table 7 summarize the distribution of age and gender, respectively. Overall, the pool covered a balanced mix of *native*, *near-native*, and *non-native* speakers, with most participants being native or near-native and reporting high confidence (4–5).

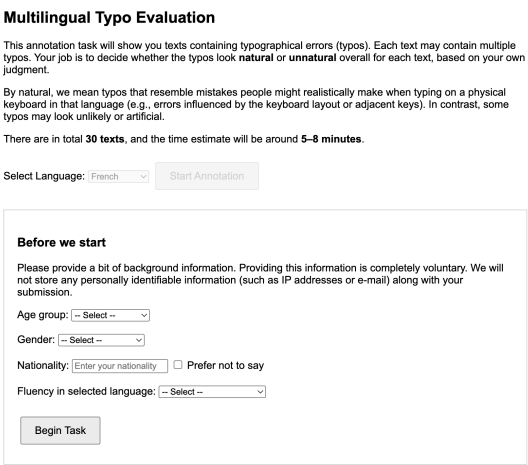

Figure 9: Screenshot of the annotation interface.

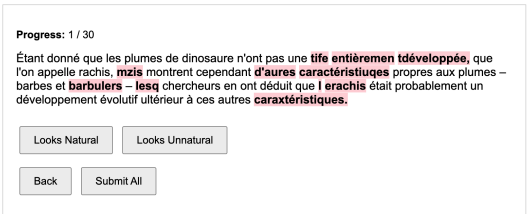

Figure 10: Example of annotating one sentence.

Findings. Extending on the results reported in §3.2, we also observe a clear effect of corruption level (cf. Figure 11): across all languages, higher typo rates lead to substantially lower "naturalness" judgments, aligning with the intuition that dense error patterns are less plausible as real-world human mistakes. Importantly, we observe that even under severe corruption (i.e., 40% and 70%), MULTYPO maintains a naturalness advantage over the naive baseline, highlighting that modeling human-typing behavior provides a better simulation of real-world typos.

C Details of Downstream Tasks

C.1 Dataset Statistics

Language Coverage MULTYPO supports 12 languages at the current stage. Each downstream task

[6] However, providing this information is completely voluntary. We did not store any personally identifiable information except for fluency (for ensuring the quality).

[7] https://www.prolific.com/

[8] For each language, we enabled Prolific's auto-filtering feature, which excluded responses completed in under 3 minutes as likely invalid.

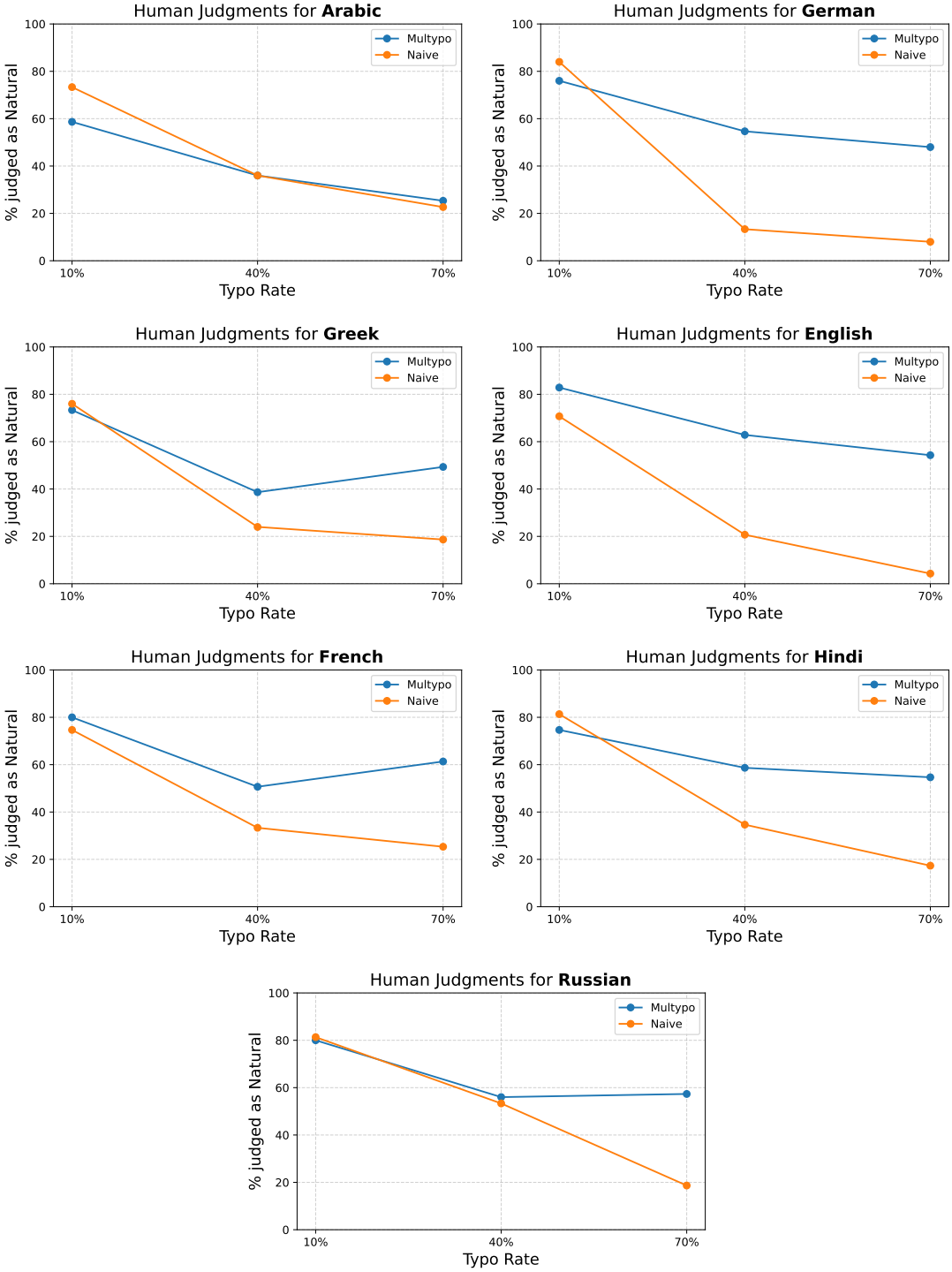

Figure 11: Human evaluation results grouped by the three considered typo rates (10%, 40%, 70%) across seven languages. Across all languages, higher typo rates reduce perceived naturalness, yet MULTYPO consistently yields more human-like typos than the naive baseline, even under higher corruption rates.

Language	Native	Near-native	Non-native	Avg. Confidence (1–5)
Arabic	7	4	4	4.1
German	6	6	3	4.0
Greek	8	3	4	3.8
English	10	13	5	3.9
French	7	4	4	3.9
Hindi	8	3	4	4.0
Russian	7	4	4	4.0

Table 5: Participant language fluency by language: number of native/near-native (very fluent)/non-native (basic) speakers and average self-reported confidence.

Language	18–24	25–34	35–44	45–54	55+	Prefer not
Arabic	5	6	3	1	–	–
German	1	1	3	3	1	6
Greek	4	4	5	–	1	1
English	6	9	2	2	3	6
French	2	6	4	3	–	–
Hindi	3	9	2	1	–	–
Russian	1	2	8	2	2	–

Table 6: Annotator age distribution across languages. Dashes indicate that no participant reported the corresponding category.

covers a slightly different set of languages, and therefore, we only evaluate on languages that are supported by MULTYPO for each dataset. Table 8 presents the languages supported in each dataset.

Instance Selection To ensure a fair and balanced evaluation, we cap each dataset at a maximum of 1,000 instances across languages where possible. Belebele contains roughly 900 instances by design and requires no further reduction. For Flores200, we selected 500 instances per translating direction (translating into and from English), yielding 1,000 prompts in total. MGSM and its Arabic and Hindi adaptations are limited to 250 examples each to maintain consistency. For XNLI and MMMLU, we subsample 1,000 instances while preserving original label (XNLI) and subject-area (MMMLU) distributions. The sampling is consistently applied in each parallel dataset to ensure comparability across languages.

C.2 Prompt Templates

For each dataset, prompts are constructed in a standardized format to ensure consistency across experiments. If typographical errors are injected, only the instances of the dataset, denoted in curly brackets {} ({language} in Flores200 is an exception), are affected. All other components of the prompt remain unchanged. The prompt templates are shown as follows.

Language	Male	Female	Prefer not	Unspecified
Arabic	6	9	–	–
German	4	5	6	–
Greek	8	6	1	–
English	9	13	6	–
French	8	5	–	2
Hindi	7	8	–	–
Russian	6	8	–	1

Table 7: Annotator gender distribution across languages. "Unspecified" refers to missing or null responses.

XNLI

Classify the relationship between the premise and hypothesis as (0) Entailment, (1) Neutral, or (2) Contradiction.
Premise: {premise}
Hypothesis: {hypothesis}
Label:

Belebele

Given the following passage, query, and answer choices, output the letter of the correct answer.
###
Passage:
{flores_passage}
###
Query:
{question}
###
Choices:
(A) {mc_answer1}
(B) {mc_answer2}
(C) {mc_answer3}
(D) {mc_answer4}
###
Answer:

MMMLU

The following are multiple choice questions (with answers) about {Subject}.
{Question}
(A) {A} (B) {B} (C) {C} (D) {D}
Answer:

MGSM

Question: {question} Let's think step by step.
Step-by-Step Answer:

Flores200: {language} → English

Translate the following sentence from {language} to English.
{language}: <BOS>{sentence}<EOS>
English: <BOS>

Flores200: English → {language}

Translate the following sentence from English to {language}.
English: <BOS>{sentence}<EOS>
{language}: <BOS>

Few-Shot Setup For each shot setting, we sample a fixed set of support examples per language and introduce typographical noise according to the

Dataset	ara_Arab	ben_Beng	deu_Latn	ell_Grek	eng_Latn	fra_Latn	heb_Hebr	hin_Deva	hye_Armn	kat_Geor	rus_Cyrl	tam_Taml
XNLI	x		x	x	x	x		x			x	
Belebele	x	x	x	x	x	x	x	x	x	x	x	x
MMMLU	x	x	x		x	x		x				
MGSM	x	x	x		x	x		x			x	
FLORES200	x	x	x	x	x	x	x	x	x	x	x	x

Table 8: Supported languages for each dataset in our evaluation.

Task	10%	40%	70%
XNLI	0.665	0.621	0.768
Belebele	0.001***	0.029*	0.000***
MMMLU	0.133	0.170	0.168
MGSM	0.002**	0.001**	0.001***
Flores200	0.213	0.071	0.000***

Table 9: Significance of performance differences between MULTYPO and *random baseline* under 3-shot setup. Each cell shows the p-value with significance stars (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$).

specified corruption level. To preserve a consistent supervision signal, the correct answers in the exemplars are left unaltered, allowing the model to learn correct associations even under noisy contexts. The same exemplars are used across all evaluations to ensure comparability. Whenever possible, examples are drawn from the training or development splits; otherwise, the first instances from the test split are selected. The remaining data points serve as independent queries during evaluation.

D Performance Comparison with Random Typo Baseline

We examine performance differences when input text is perturbed by MULTYPO – which simulates human typing behavior – versus a *random typo baseline* that applies the same four operations described in §3.1 but disregards keyboard layout constraints. For this evaluation, we use `Qwen3-4B` (instruction-tuned) across all five tasks.

As shown in Figure 12, performance under random perturbations is consistently lower than with MULTYPO. This suggests that models are more robust to MULTYPO typos, likely because they better approximate realistic human typing errors, some of which may already be represented in pretraining corpora. Table 9 further confirms these trends: (1) For natural language understanding tasks (e.g., XNLI), the performance gap is small and not statistically significant. (2) For generation tasks – particularly those requiring reasoning – the random baseline leads to significantly larger degradation compared to MULTYPO.

E Complementary Results on Translation

We also compare the performance of **Qwen** and **OLMo** models when translating *from English* vs. *into English*, as shown in Figure 13 and Figure 14. In general, we observe the same trend as in Figure 7. That is, translations *from* English are more robust than those *into* English. This trend is typically noticeable when involving low-resource languages that are written in non-Latin scripts, such as hye_Armn and kat_Geor. To sum up, these findings further support our claim that typos in lower-resource input languages might severely impair the understanding and, therefore, result in bad translation quality.

F Experimental Environment and Hyperparameters

All experiments are conducted on NVIDIA RTX A6000 GPUs. We use vLLM to process the prompts and obtain the response for each prompt.[9] The default sampling parameters (top-k, top-p, etc.) of vLLM are used. We set the different maximum generation tokens for each dataset: 5 for XNLI, 100 for Belebele, 5 for MMMLU, 200 for MGSM, and 100 for Flores200, based on preliminary experimental results.

[9] https://docs.vllm.ai/en/v0.7.3/

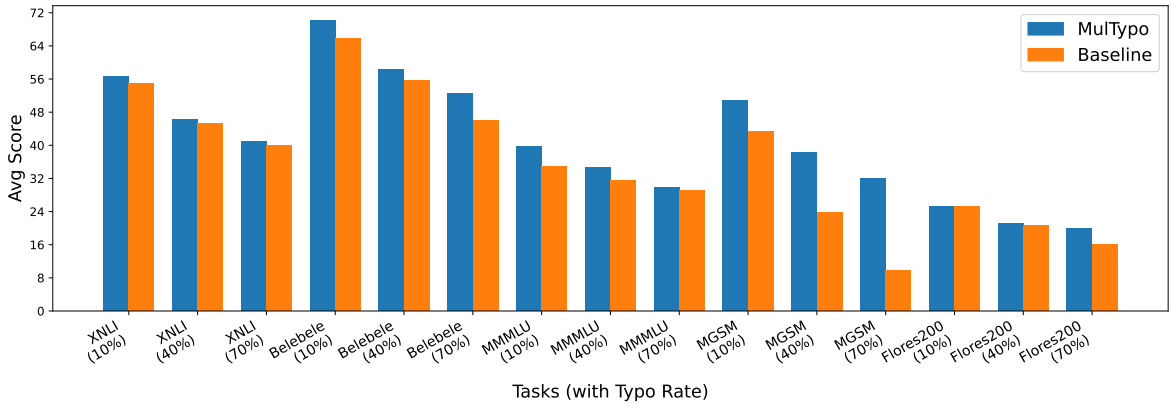

Figure 12: Performance comparison of MULTYPO and random baseline under different typo rates (10%, 40%, 70%) in the 3-shot setting. Bars show average performance across all languages for each task.

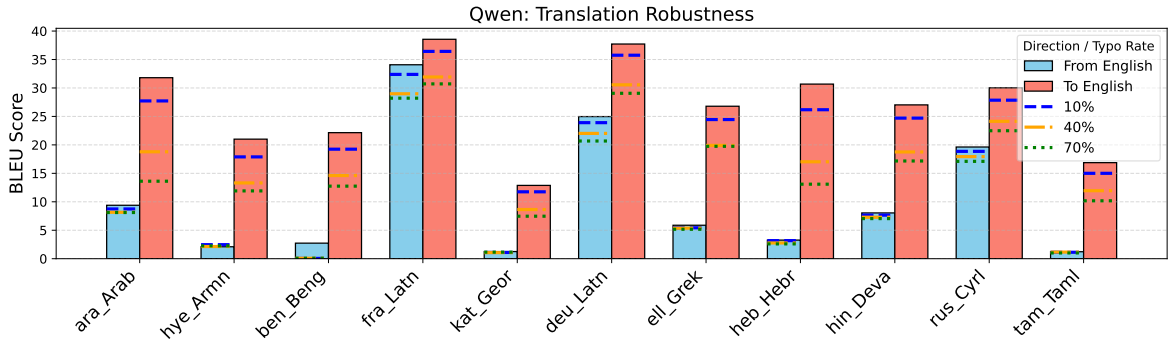

Figure 13: Robustness of **Qwen** models on **Flores200** under different levels of typographical noise. Translation from English seems to be more robust compared to translation to English.

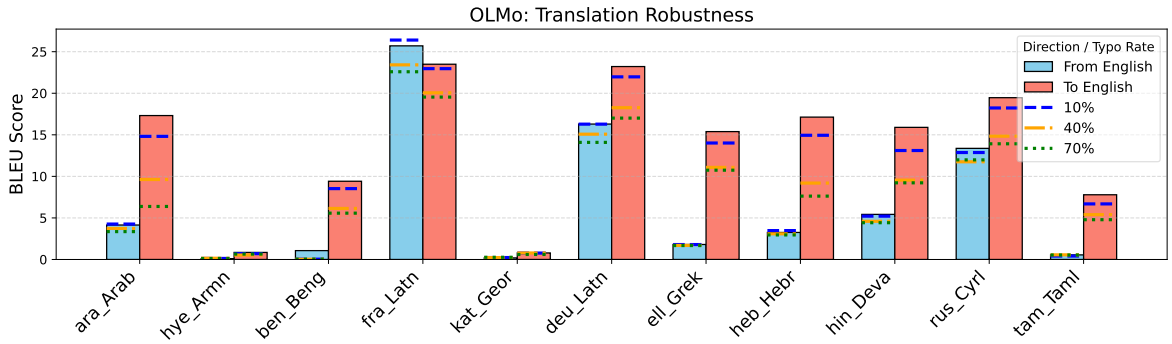

Figure 14: Robustness of **OLMo** models on **Flores200** under different levels of typographical noise. Translation from English seems to be more robust compared to translation to English.