Enhancing Text Editing for Grammatical Error Correction: Arabic as a Case Study

Bashar Alhafni[†] and **Nizar Habash**
Computational Approaches to Modeling Language Lab
New York University Abu Dhabi
[†]Mohamed bin Zayed University of Artificial Intelligence
{alhafni,nizar.habash}@nyu.edu

Abstract

Text editing frames grammatical error correction (GEC) as a sequence tagging problem, where edit tags are assigned to input tokens, and applying these edits results in the corrected text. This approach has gained attention for its efficiency and interpretability. However, while extensively explored for English, text editing remains largely underexplored for morphologically rich languages like Arabic. In this paper, we introduce a text editing approach that derives edit tags directly from data, eliminating the need for language-specific edits. We demonstrate its effectiveness on Arabic, a diglossic and morphologically rich language, and investigate the impact of different edit representations on model performance. Our approach achieves SOTA results on two Arabic GEC benchmarks and performs on par with SOTA on two others. Additionally, our models are over six times faster than existing Arabic GEC systems, making our approach more practical for real-world applications. Finally, we explore ensemble models, demonstrating how combining different models leads to further performance improvements. We make our code, data, and pretrained models publicly available.[1]

1 Introduction

Grammatical Error Correction (GEC) is a well-studied problem, particularly in English, with numerous datasets and shared tasks (Ng et al., 2013, 2014; Bryant et al., 2019). GEC has applications in both writing assistance for native speakers (L1) and language learning for second-language (L2) learners. While neural machine translation (NMT) approaches have long dominated GEC and continue to achieve strong results when trained on large amounts of data (Stahlberg and Kumar, 2024; Bryant et al., 2023), they are not inherently the most efficient. Unlike MT, where input and output sequences differ significantly, GEC typically involves minimal changes, with most input tokens copied to the output. Employing full-sequence autoregressive models in such cases can be computationally wasteful (Stahlberg and Kumar, 2020).

A highly efficient and competitive alternative to sequence-to-sequence (Seq2Seq) models is text editing, which frames GEC as a sequence tagging problem. Instead of generating text autoregressively, text editing models assign edit labels to input tokens, leading to a more efficient and interpretable corrections. However, most popular text editing approaches require effort to design language-specific edit tag sets (Awasthi et al., 2019; Omelianchuk et al., 2020; Mesham et al., 2023). This limits their adaptability for morphologically rich languages like Arabic (Kwon et al., 2023), where the space of possible edits is large.

Inspired by recent advancements in text editing (Awasthi et al., 2019; Malmi et al., 2019; Omelianchuk et al., 2020; Straka et al., 2021; Mesham et al., 2023), we introduce a novel text editing approach that eliminates the need for language-specific edits. Instead, our method derives edit tags directly from data, making it more adaptable and scalable across different linguistic settings. We demonstrate the effectiveness of our approach on Arabic GEC. Our contributions are as follows:

1. We introduce the first successful application of text editing to Arabic GEC and study the effect of edit representation on the task.
2. We achieve SOTA results on two Arabic GEC benchmarks and perform on par with SOTA on two others.
3. Our models are over six times faster than existing Arabic GEC systems, making them more practical for real-world applications.
4. We show through ensembling experiments how different models complement each other, leading to significant performance gains.

[1] https://github.com/CAMeL-Lab/text-editing

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 17892–17914

2 Background and Related Work

2.1 Grammatical Error Correction

GEC has been approached using a variety of methods, with Transformer-based systems being the most popular (Bryant et al., 2023). The use of Transformer-based architectures in GEC began by framing the task as a neural machine translation (NMT) problem (Junczys-Dowmunt et al., 2018; Yuan et al., 2019; Zhao et al., 2019; Grundkiewicz et al., 2019; Katsumata and Komachi, 2020; Kaneko et al., 2020; Wan et al., 2020; Yuan et al., 2021; Yuan and Bryant, 2021; Stahlberg and Kumar, 2021; Rothe et al., 2021; Zhou et al., 2023; Luhtaru et al., 2024).

To improve efficiency and interpretability, text editing models have emerged as an alternative to Seq2Seq approaches (Awasthi et al., 2019; Malmi et al., 2019; Stahlberg and Kumar, 2020; Mallinson et al., 2020; Omelianchuk et al., 2020; Straka et al., 2021; Mallinson et al., 2022; Tarnavskyi et al., 2022; Mesham et al., 2023; Zhang et al., 2023). Unlike Seq2Seq models, which generate corrected text from scratch, text editing models treat GEC as a sequence tagging task, producing a set of edit operations that modify the erroneous input. Our work follows this text editing paradigm.

LLMs have also been evaluated on GEC (Fang et al., 2023; Coyne et al., 2023; Wu et al., 2023; Loem et al., 2023; Raheja et al., 2023; Kaneko and Okazaki, 2023; Raheja et al., 2024; Davis et al., 2024; Katinskaia and Yangarber, 2024; Omelianchuk et al., 2024; Mita et al., 2024; Kaneko and Okazaki, 2024). However, despite their strong generalization capabilities, they remain less effective than Seq2Seq and text editing models.

2.2 Arabic Grammatical Error Correction

Arabic exhibits a diglossic (Ferguson, 1959) linguistic nature where a non-standard variety, Dialectal Arabic (DA), coexists with Modern Standard Arabic (MSA), the standard form of the language.

MSA GEC The first major efforts on MSA GEC were initiated by the Qatar Arabic Language Bank (QALB) project (Zaghouani et al., 2014, 2015), which organized the QALB-2014 (Mohit et al., 2014) and QALB-2015 (Rozovskaya et al., 2015) shared tasks. More recently, Habash and Palfreyman (2022) introduced the ZAEBUC corpus, a dataset of essays written by native Arabic-speaking university students. Approaches to MSA GEC have included feature-based classifiers (Rozovskaya et al., 2014; Farra et al., 2014; Bougares and Bouamor, 2015; Nawar, 2015) and NMT-based systems (Watson et al., 2018; Solyman et al., 2021, 2022, 2023). LLMs have also been evaluated for MSA GEC (Kwon et al., 2023; Alhafni et al., 2023; Magdy et al., 2024), but attempts to adapt text editing models have been largely ineffective. The current SOTA was established by Alhafni et al. (2023), who incorporated contextualized morphological preprocessing and grammatical error detection (GED) features into Seq2Seq models, achieving SOTA results on the QALB-2014, QALB-2015, and ZAEBUC datasets.

DA GEC Dialectal Arabic (DA) comprises multiple regional varieties that differ from MSA and each other in phonology, morphology, and lexicon. While primarily spoken, DA lacks standardized orthography, though its written use has grown on social media, where it appears in varied and noisy forms. To address this, Habash et al. (2012a, 2018) introduced the Conventional Orthography for Dialectal Arabic (CODA), a standardized spelling convention for DA. CODA has since been used to develop multiple DA datasets (Habash et al., 2012b; Eskander et al., 2013; Maamouri et al., 2014; Diab et al., 2014; Pasha et al., 2014; Jarrar et al., 2016; Khalifa et al., 2018). Building on this work, Eryani et al. (2020) created the MADAR CODA Corpus, which consists of parallel sentences in CODA and their original raw form for five Arabic city dialects. CODAfication–the process of normalizing DA into CODA–has been addressed using feature-based methods (Eskander et al., 2013) and morphological disambiguation models (Pasha et al., 2014; Zalmout et al., 2018; Khalifa et al., 2020; Zalmout and Habash, 2020; Obeid et al., 2022). More recently, Alhafni et al. (2024) framed CODAfication as a DA GEC problem, benchmarking pretrained Arabic Seq2Seq models on the MADAR CODA corpus and demonstrating that incorporating dialect identification improves performance.

In this work, we propose a generalizable and efficient text editing approach and evaluate its effectiveness on both MSA and DA GEC. For MSA GEC, we benchmark our models against Alhafni et al. (2023) on QALB-2014, QALB-2015, and ZAEBUC. For DA GEC, we build on Alhafni et al. (2024) by framing CODAfication as a DA GEC problem, evaluating our approach on the MADAR CODA corpus and comparing it to their results.

Corrected	النفسية . *Alnfsyh*		الصحة *AlSHh*			سيما *symA*	ولا *wlA*	بالصحة *bAlSHh*			الاهتمام *AlAhtmAm*		يجب *yjb*		(a)
	7		6		5	4		3		2	1		0		
Erroneous	النفسيه *Alnfsyh*		الصحه *AlSHh*		في *fy*	ولاسيما *wlAsymA*		لصحه *lSHh*		ب *b*	الإهتمام *AlĂhtmAm*		يجب *yjb*		(b)
Word Edits	KKKKKKR_[ة]A_[.]		KKKKR_[ة]		DD	KKKI_[ ]KKKK		MI_[ا]KKKR_[ة]		K	KKR_[ا]KKKKK		KKK		(c)
Word Edits (Compressed)	K*R_[ة]A_[.]		K*R_[ة]		D*	KKKI_[ ]K*		MI_[ا]K*R_[ة]		K*	KKR_[ا]K*		K*		(d)
	7b	7a	6b	6a	5	4		3b	3a	2	1b	1a	0		
Tokenized Erroneous	ه## *##h*	النفسي *Ălnfsy*	ه## *##h*	الصح *AlSH*	في *fy*	ولاسيما *wlAsymA*		ه## *##h*	لصح *lSH*	ب	هتمام## *##htmAm*	الإ *AlĂ*	يجب *yjb*		(e)
Subword Edits	R_[ة]A_[.]	KKKKKK	R_[ة]	KKKK	DD	KKKI_[ ]KKKK		R_[ة]	MI_[ا]KKK	K	KKKKK	KKR_[ا]	KKK		(f)
Subword Edits (Compressed)	R_[ة]A_[.]	K*	R_[ة]	K*	D*	KKKI_[ ]K*		R_[ة]	MI_[ا]K*	K*	K*	K*R_[ا]	K*		(g)

Figure 1: An example showing the different edit representations: words, words (compressed), subwords, and subwords (compressed). The edit operations are keep (**K/K***), delete (**D/D***), merge before (**M**), replace (**R_[c]**), insert (**I_[c]**), and append (**A_[c]**). Solid lines indicate word alignments between the corrected and erroneous sentences, while dotted lines denote erroneous subword boundaries. The sentence in the figure can be translated as *"Health, especially mental health, must be taken care of"*.

3 Approach

We adopt a text editing approach to GEC and frame the task as a sequence tagging problem. Formally, given an input erroneous sequence $x = x_1, x_2, ..., x_n$, the goal is to assign a sequence of edit operations $e = e_1, e_2, ..., e_n$; $e_i \in E$, where E is the edit vocabulary, such that applying edit e_i on the input token x_i at each position i would result in the corrected sequence $y = y_1, y_2, ..., y_m$. In the next two sections, we describe how we extract the edits and the edit representations we use to build our edit-based taggers.

3.1 Edit Extraction

We begin by aligning erroneous and corrected sentence pairs at the word level using a weighted Levenshtein edit distance (Levenshtein, 1966), which represents the minimum number of insertions, deletions, and replacements required to correct the erroneous sentence, with each edit affecting a single word. However, some errors span multiple words. To capture multi-word edits, we follow the approach of Alhafni et al. (2023) by extending the alignment process with an iterative algorithm that greedily merges or splits adjacent words, minimizing the overall cumulative edit distance. After obtaining the word-level alignment, we apply the algorithm again, this time to each aligned word pair rather than the entire sentence, to determine character-level alignments. This process identifies the minimal character edits in terms of keep (**K**), delete (**D**), merge before (**M**), insert (**I_[c]**), and replace (**R_[c]**) that are needed to transform each erroneous word into its correction, where the inserted or replaced character (**c**) is explicitly specified.

Figure 1 presents an example of an aligned erroneous-corrected sentence pair along with the corresponding edits. For instance, in row b, the erroneous word الإهتمام *AlĂhtmAm*[2] (word 1) requires the edit KKR_[ا]KKKKK (row c) which consists of eight character edits–one replacement and seven keeps–to produce its corrected form الاهتمام *AlAhtmAm*. Similarly, لصحه *lSHh* (row b, word 3), must be merged with the word before it, in addition to one insertion and one replacement (MI_[ا]KKKR_[ة], row c).

In some cases, corrections require the insertion of entirely new characters, forming additional words in the erroneous input. Since we frame the task as a sequence tagging problem, we represent these insertions as appends (**A_[c]**) to existing edits rather than introducing standalone edits. This ensures that all edits, including word insertions, remain within the tagging framework. For example, to insert a period at the end of the erroneous sentence in Figure 1, we append the tag (A_[.]) to the edit of the final word (row c, word 7).

3.2 Edit Representation

The edit representation directly influences the size of the edit vocabulary ($|E|$), creating an impor-

[2]Arabic HSB transliteration (Habash et al., 2007).

tant trade-off: a larger vocabulary offers more precise corrections but increases model complexity, whereas a smaller vocabulary enhances learning efficiency at the cost of expressiveness. Controlling $|E|$ is crucial to avoid the explosion of possible edits, which is particularly important when working with morphologically rich languages like Arabic. We explore four methods for controlling $|E|$ while maintaining sufficient coverage.

Edit Compression Once we obtain character-level edits for each word, we compress them into a more compact representation. The motivation behind this transformation is that while different words may undergo the same type of correction, their character-level edits can differ due to variations in word length. For example, in row b of Figure 1, both words 0 and 2 share a keep edit, yet they receive different edit labels because of their length differences (row c). To address this, we introduce a generalized notation for common edit patterns. Consecutive keep (**K**) and delete (**D**) operations are represented as **K*** and **D***, respectively. Similarly, consecutive insertions and appends are merged into a single operation, represented as **I_[c*]** for insertions and **A_[c*]** for appends, indicating the insertion or appending of multiple characters.

Since there are multiple ways to compress an edit sequence, we select the optimal strategy based on the frequency distribution of edit patterns in the training data. This approach ensures that the most common transformations are encoded in a way that balances expressiveness with efficiency, resulting in a more structured and learnable edit representation.

Input Unit Since Transformer-based models operate at the subword level, we project character-level edits onto subwords while maintaining their boundaries to ensure proper alignment. This not only ensures consistency with the model's input representation but also helps reduce the edit vocabulary size. Our approach is inspired by the method of Straka et al. (2021), but it differs in several key aspects: (1) Straka et al. (2021) tokenize the erroneous and corrected sentence pairs before aligning them to extract the edits at the subword level. In contrast, our method extracts edits at the word level and then projects them onto subwords; (2) They limit the number of character-level edits per subword edit, while our approach imposes no such restrictions, allowing for broader coverage.

Input	Comp.	Subset	Prune	Edits	OOV%	$F_{0.5}$
Word	✗	All	-	16,221	1.00%	98.4
Subword	✗	All	-	9,060	0.36%	98.7
Word	✓	All	-	10,410	1.00%	98.4
Subword	✓	All	-	6,170	0.36%	98.7
Subword	✓	NoPnx	-	4,799	0.27%	98.8
Subword	✓	Pnx	-	160	0.01%	99.4
Subword	✓	All	10	683	0.75%	98.1
Subword	✓	All	20	442	1.02%	97.7
Subword	✓	All	30	329	1.24%	97.4
Subword	✓	NoPnx	10	520	0.56%	98.2
Subword	✓	NoPnx	20	335	0.75%	97.8
Subword	✓	NoPnx	30	250	0.92%	97.5
Subword	✓	Pnx	10	48	0.02%	99.4
Subword	✓	Pnx	20	35	0.05%	99.4
Subword	✓	Pnx	30	29	0.05%	99.3

Table 1: Edit statistics on QALB-2014. **Input** is the input unit (word or subword). **Comp.** indicates whether the edit is compressed. **Subset** specifies whether the edits capture all errors, punctuation-only errors (Pnx), or non-punctuation errors (NoPnx). **Edits** represents the total number of unique edits in the training set. **OOV%** is the percentage of out-of-vocabulary edits (non-unique) in the Dev set of QALB-2014.

Figure 1 presents the subword-level edits in both their uncompressed (row f) and compressed (row g) forms. In the uncompressed subword-level edits, we observe that two subwords (3b and 6b in row e), which belong to different words, share the same edit (R_[ð]). In the compressed representation, we notice that several subwords–such as 0, 1b, 2, 6a, and 7a–end up sharing the same edit (K*).

Edit Segregation Both the MSA GEC datasets we report on, QALB-2014 and ZAEBUC, exhibit high frequencies of punctuation errors, with punctuation accounting for 40% of the errors in QALB-2014 and 15% in ZAEBUC training sets (Alhafni et al., 2023). To reduce the number of edits that the MSA GEC models must learn, we *segregate* punctuation edits from non-punctuation edits. This results in two versions of the data: one where only non-punctuation errors are tagged, and another where all non-punctuation errors are corrected, leaving only punctuation errors for the model to focus on. Note that this separation is applied only to the MSA GEC datasets we report on, and not to the DA GEC dataset. Additionally, this approach requires training two systems to be applied sequentially during inference: the first system fixes non-punctuation errors, while the second system addresses only punctuation errors.

Edit Pruning Morphologically rich languages, in particular, tend to have many infrequent edits in GEC datasets. To improve the model's learning ability, we analyze the distribution of edits in the training data and prune those that occur less frequently than a threshold T, replacing them with the "keep" edit. This pruning is applied exclusively during training, enabling the model to focus on frequent and informative edits.

3.3 Edit Coverage

Table 1 presents edit statistics for QALB-2014, illustrating the impact of our strategies to reduce the edit vocabulary size $|E|$ on edit coverage and upper-bound (oracle) performance on the development (Dev) set. Edit coverage measures the proportion of training edits found in the Dev set, while oracle performance is evaluated using the MaxMatch (M^2) scorer (Dahlmeier and Ng, 2012) $F_{0.5}$ (§4.2). We use AraBERTv02 (Antoun et al., 2020) for subword tokenization, as it yielded the best results among our tested models (more details in §5).

Switching from word-level to subword-level edits reduces unique training edits by 44% (16,221 to 9,060) and lowers the Dev set OOV rate from 1% to 0.4%, yielding a 0.3-point $F_{0.5}$ gain. Edit compression further reduces unique edits while preserving OOV% and oracle performance.

Segregating punctuation (Pnx) from non-punctuation (NoPnx) edits reduces combined training edits (4,799+160 from 6,170). However, NoPnx results are not directly comparable, as punctuation is explicitly removed before the evaluation. Pnx $F_{0.5}$ scores are higher as they are evaluated on a Dev set with non-punctuation errors already corrected, making the test easier.

To assess the impact of pruning, we apply frequency thresholds of 10, 20, and 30 to remove low-frequency edits. As expected, pruning reduces the number of unique training edits and increases the OOV% in the Dev set, yet $F_{0.5}$ remains largely unaffected. This suggests that the majority of the 6,170 compressed subword edits occur infrequently and contribute little to the model's upper-bound performance. A similar trend is observed for both Pnx and NoPnx edits, reinforcing the idea that many low-frequency edits can be pruned without degrading oracle performance.

We present the same analysis on all datasets in Appendix B Table 9.

Dataset	Split	Lines	Words	Err.%	Domain
QALB-2014	Train	19K	1M	30%	Comments
	Dev	1K	54K	31%	Comments
	Test	968	51K	32%	Comments
QALB-2015	Test	920	49K	27%	Comments
ZAEBUC	Train	150	25K	24%	Essays
	Dev	33	5K	25%	Essays
	Test	31	5K	26%	Essays
MADAR CODA	Train	7K	40K	22%	Comments
	Dev	1.5K	9K	20%	Comments
	Test	1.5K	9K	21%	Comments

Table 2: Corpus statistics of MSA (QALB, ZAEBUC) and DA (MADAR CODA) GEC datasets.

4 Experimental Setup

4.1 Data

MSA GEC We report on three publicly available MSA GEC datasets. The first is the QALB-2014 shared task dataset (Mohit et al., 2014), followed by the native (L1) test set from the QALB-2015 shared task (Rozovskaya et al., 2015). The third dataset is ZAEBUC (Habash and Palfreyman, 2022). QALB-2014 and the L1 test set of QALB-2015 contain comments by native speakers from the Aljazeera news website, whereas ZAEBUC consists of essays written by native university students. We use the publicly available splits for QALB-2014 and QALB-2015, while for ZAEBUC, we use the splits created by Alhafni et al. (2023).

DA GEC We use the MADAR CODA corpus (Eryani et al., 2020), a set of 10,000 sentences from five Arabic city dialects (Beirut, Cairo, Doha, Rabat, and Tunis) written in the CODA standard in parallel with their original raw form. The sentences come from the Multi-Arabic Dialect Applications and Resources (MADAR) Project (Bouamor et al., 2018) and are in parallel across the cities (2,000 sentences per city). We use the publicly available splits created by Alhafni et al. (2024).

Table 2 summarizes the dataset statistics.

4.2 Evaluation

We use the MaxMatch (M^2) scorer (Dahlmeier and Ng, 2012), which evaluates GEC systems by comparing hypothesis edits with reference edits, calculating precision (P), recall (R), F_1, and $F_{0.5}$ scores. $F_{0.5}$ weighs precision twice as much as recall, to prioritize the accuracy of edits relative to all edits made by the system.

	QALB-2014				ZAEBUC			
	P	**R**	**F_1**	**$F_{0.5}$**	**P**	**R**	**F_1**	**$F_{0.5}$**
A'2023 (Seq2Seq)	83.2	64.9	72.9	78.7	87.3	70.6	78.1	83.4
A'2023 (Seq2Seq++)	83.1	67.9	74.7	79.6	87.6	73.9	80.2	84.5
GPT-3.5-turbo	68.6	58.6	63.2	66.3	71.0	63.5	67.1	69.4
GPT-4o	80.7	65.7	72.4	77.2	86.5	**76.8**	**81.3**	84.3
Fanar	69.7	63.7	66.6	68.4	76.3	73.6	74.9	75.8
Jais-13B-Chat	49.1	36.9	42.1	46.0	50.2	19.7	28.3	38.4
SWEET	81.8	68.8	74.7	78.8	85.8	72.3	78.4	82.7
SWEET^2	81.9	**70.4**	75.7	79.3	85.8	73.3	79.1	83.0
$\text{SWEET}^2_{\text{NoPnx}}$ + $\text{SWEET}_{\text{Pnx}}$	83.7	68.8	75.6	80.3†	86.7	73.9	79.8	83.8
3-Ensemble	84.9	68.8	**76.0**	81.1	89.6	72.8	80.3	85.6
4-Ensemble	**89.1**	61.6	72.8	**81.8**‡	**93.3**	68.3	78.9	**86.9**‡

Table 3: MSA GEC results on the Dev sets of QALB-2014 and ZAEBUC. A'2023 is Alhafni et al. (2023). Best non-ensemble results are underlined; best overall results are in bold. † denotes statistical significance over the best baseline; ‡ denotes statistical significance over both the best baseline and the best non-ensemble model.

	P	**R**	**F_1**	**$F_{0.5}$**
A'2024 (Seq2Seq)	86.8	77.4	81.8	84.7
A'2024 (Seq2Seq++)	87.6	**79.3**	83.3	85.8
GPT-3.5-turbo	35.5	29.7	32.3	34.1
GPT-4o	53.7	54.4	54.1	53.8
Fanar	24.5	28.8	26.4	25.2
Jais-13B-Chat	14.1	15.0	14.5	14.3
SWEET	89.1	75.5	81.7	86.0
SWEET^2	87.5	73.5	79.9	84.3
3-Ensemble	91.7	77.4	**83.9**	88.4
4-Ensemble	**93.8**	72.5	81.8	**88.6**‡

Table 4: DA GEC results on the MADAR CODA Dev set. A'2024 is Alhafni et al. (2024). Best non-ensemble results are underlined; best overall results are in bold. ‡ denotes statistical significance over both the best baseline and the best non-ensemble model.

4.3 Models

LLMs We evaluate four LLMs: two commercial models and two open-source, Arabic-centric models. The commercial models include OpenAI's GPT-3.5-turbo and GPT-4o (OpenAI et al., 2024), while the Arabic-centric models are Jais-13B-Chat (Sengupta et al., 2023) and the recently introduced Fanar LLM (Team et al., 2025). We prompt GPT-3.5-turbo, GPT-4o, and Fanar through the OpenAI API, while Jais-13B-Chat is prompted using Hugging Face's Transformers (Wolf et al., 2020). Our experiments use both English and Arabic prompts, employing 0-shot and 5-shot prompting strategies. We design the prompts to elicit minimal edit-style corrections, ensuring that the LLMs' outputs remain as close as possible to the original input in phrasing and lexical choices. We present our prompts in Figures 2 and 3 in Appendix H.

Edit Taggers To investigate the impact of edit representation design on performance (§3.2), we build edit taggers with different configurations. For word-level tagging, we use the representation of the first subword of each word and pass it through the subsequent layers. For subword-level tagging, we use the representation of each subword individually. Several Arabic pretrained transformer encoders based on BERT (Devlin et al., 2019) have been developed (Antoun et al., 2020; Abdul-Mageed et al., 2021; Inoue et al., 2021; Ghaddar et al., 2022). We select the three best-performing Arabic BERT models, as identified by Inoue et al. (2021) across various sentence and token classification tasks: AraBERTv02 (Antoun et al., 2020), ARBERTv2 (Abdul-Mageed et al., 2021), and CAMeLBERT-MSA (Inoue et al., 2021).

For QALB-2014, our edit taggers are trained exclusively on QALB-2014, following the shared task restrictions. For QALB-2015 (L1), we train only on QALB-2014 for consistency. For ZAEBUC, we train on both QALB-2014 and ZAEBUC, upsampling ZAEBUC tenfold to address its smaller size and domain shift. For DA GEC, we train on the MADAR CODA training split. The hyperparameters we used are detailed in Appendix A.

4.4 Ensembling

We construct majority vote ensemble models by aggregating the outputs of multiple GEC systems.

	QALB-2014				QALB-2015				ZAEBUC			
	P	**R**	**F_1**	**$F_{0.5}$**	**P**	**R**	**F_1**	**$F_{0.5}$**	**P**	**R**	**F_1**	**$F_{0.5}$**
A'2023 (Seq2Seq)	84.0	64.7	73.1	79.3	82.0	71.7	76.5	79.7	86.0	71.6	78.2	82.7
A'2023 (Seq2Seq++)	84.2	65.4	73.6	79.6	82.6	72.1	77.0	80.3	85.9	73.4	79.2	83.1
GPT-4o	81.5	65.5	72.6	77.7	81.1	**74.3**	77.5	79.6	84.4	**75.9**	79.9	82.5
SWEET^2	82.6	**69.5**	**75.5**	79.6	80.0	**74.3**	77.0	78.8	85.5	74.4	79.6	83.0
SWEET$^2_{\text{NoPnx}}$ + SWEET$_{\text{Pnx}}$	84.5	67.7	75.2	80.5†	82.2	73.6	77.7	80.3	85.7	74.1	79.5	83.1
3-Ensemble	85.7	67.4	75.4	81.3	83.7	73.3	**78.1**	81.3	89.7	73.7	**80.9**	85.9
4-Ensemble	**89.7**	60.2	72.0	**81.7**‡	**88.3**	66.7	76.0	**82.9**‡	**93.4**	68.9	79.3	**87.2**‡

Table 5: MSA GEC results on the Test sets of QALB-2014, QALB-2015 (L1), and ZAEBUC. A'2023 is Alhafni et al. (2023). Best non-ensemble results are underlined; best overall results are in bold. † denotes statistical significance over the best baseline; ‡ denotes statistical significance over both the best baseline and the best non-ensemble model.

	P	**R**	**F_1**	**$F_{0.5}$**
A'2024 (Seq2Seq)	87.3	78.0	82.4	85.2
A'2024 (Seq2Seq++)	88.4	**79.0**	83.4	86.3
GPT-4o	56.1	54.8	55.5	55.9
SWEET	89.4	76.6	82.5	86.5
3-Ensemble	92.2	77.7	**84.3**	**88.9**‡
4-Ensemble	**94.0**	72.9	82.1	88.8

Table 6: DA GEC results on the Test set of MADAR CODA. A'2024 is Alhafni et al. (2024). Best non-ensemble results are underlined, best overall results are in bold. ‡ denotes statistical significance over both the best baseline and the best non-ensemble model.

This is enabled by our edit extraction algorithm (§3.1), which allows us to align and extract edits from models with different architectures. Using this algorithm, we first align each model's output with the input text, extract the proposed edits, and then determine the final edit sequence through majority voting. Following Tarnavskyi et al. (2022), we retain an edit only if at least $k-1$ models out of k models predict it; otherwise, we leave the input unchanged. This strategy prioritizes precision over recall, which is crucial for GEC systems, as precision is generally more important than correcting every possible error (Bryant et al., 2023).

5 Results

Tables 3 and 4 show the Dev results for MSA and DA GEC, respectively. For each dataset, we compare our models with the best-performing Seq2Seq and Seq2Seq++ baselines reported by Alhafni et al. (2023) and Alhafni et al. (2024). The Seq2Seq++ setups incorporate additional signals, such as morphological preprocessing and GED information for MSA GEC, or dialect identification for DA GEC. Full results for all Seq2Seq-based baseline variants across datasets are provided in Appendices E and F.

LLMs We present LLMs results on MSA and DA GEC using their best setups, optimized for average $F_{0.5}$ across all datasets based on prompt language and strategy (0-shot vs. 5-shot). Full results are in Table 11 (Appendix D).

For QALB-2014, GPT-4o and Fanar outperform GPT-3.5 and Jais-13B-Chat, with GPT-4o achieving the best performance, though none surpass Alhafni et al. (2023). On ZAEBUC, GPT-4o leads, achieving the highest recall (76.8) and F_1 (81.3). For DA GEC, GPT-4o is the top LLM, but overall LLM performance is notably lower than for MSA.

Edit Taggers Table 10 (Appendix C) presents the full edit tagging results on the Dev sets, exploring edit design choices using CAMeLBERT-MSA, AraBERTv02, and ARBERTv2. AraBERTv02 consistently performs best. Subword-level edits, compression, and pruning improve performance, with optimal pruning thresholds of 10 for QALB-2014 and MADAR CODA, and 30 for ZAEBUC.

The optimal setup for each dataset (subword, compression, pruning) is presented in Tables 3 and 4. We henceforth refer to this system as SWEET (Subword Edit Error Tagger). SWEET achieves an $F_{0.5}$ of 78.8 on QALB-2014 and 86.0 on MADAR CODA, outperforming the Seq2Seq baseline on QALB-2014 and setting a new SOTA on MADAR CODA (though the improvement is not statistically significant).[3] On ZAEBUC, it scores 82.7 $F_{0.5}$, trailing behind the Seq2Seq baseline.

[3]Statistical significance was done using a two-sided approximate randomization test.

	QALB-2014			ZAEBUC			MADAR CODA		
	Baseline	**SWEET**	**Ensemble**	**Baseline**	**SWEET**	**Ensemble**	**Baseline**	**SWEET**	**Ensemble**
Delete	41.1	44.3	**45.2**	51.9	**63.6**	62.5	0.0	0.0	0.0
Merge-B	**94.0**	93.7	93.8	96.7	**96.9**	96.6	**94.4**	86.6	92.7
Merge-I	**93.8**	93.5	93.6	96.7	**96.9**	96.6	**93.6**	84.8	91.6
M	**33.9**	33.6	28.6	48.6	**50.0**	41.7	**82.5**	78.0	82.4
M+O	58.0	**61.0**	60.6	55.6	**100.0**	0.0	0.0	0.0	0.0
O	94.3	**94.5**	94.4	**94.4**	**94.4**	94.1	**92.1**	90.2	91.4
O+X	78.1	81.5	**83.3**	0.0	0.0	0.0	0.0	0.0	0.0
P	75.0	75.6	**76.8**	62.8	**71.9**	70.4	**65.8**	35.7	55.6
S	46.4	57.1	**57.4**	40.4	**47.6**	46.9	**83.1**	82.3	83.0
X	61.2	61.4	**62.4**	72.9	**74.1**	**74.1**	73.8	76.9	**79.5**
Split	87.1	83.8	**87.6**	88.2	90.0	**95.2**	85.9	83.3	**86.8**
UNK	**59.2**	55.0	56.0	**63.1**	44.6	47.6	93.0	**94.6**	94.1
C	**97.1**	96.4	94.7	**96.1**	96.0	93.9	**97.0**	96.0	94.7
Macro Avg.	70.7	71.6	**71.9**	66.7	**71.2**	63.1	**78.3**	73.5	77.4

Table 7: Error type performance on the Dev sets of QALB-2014, ZAEBUC, and MADAR CODA for the best Seq2Seq++ baseline, the best SWEET model ($\text{SWEET}^2_{\text{NoPnx}}$ + $\text{SWEET}_{\text{Pnx}}$ for QALB-2014 and ZAEBUC; SWEET for MADAR CODA), and the best ensemble (4-Ensemble). Results are reported in terms of $F_{0.5}$. Best non-ensemble results are underlined; best overall results are in bold. UNK refers to unknown error types; C refers to correct words.

Consistent with previous work on text editing (Omelianchuk et al., 2020; Straka et al., 2021), we find that iterative correction improves MSA GEC up to two iterations (SWEET^2), achieving 79.3 on QALB-2014 and 83.0 $F_{0.5}$ on ZAEBUC, with the highest recall on QALB-2014 (70.4). However, iterative correction degrades DA GEC performance.

Separating non-punctuation edits ($\text{SWEET}_{\text{NoPnx}}$) from punctuation edits ($\text{SWEET}_{\text{Pnx}}$) improves MSA GEC performance. The best setup applies these systems in sequence: two iterations of non-punctuation correction followed by one iteration of punctuation correction ($\text{SWEET}^2_{\text{NoPnx}}$ + $\text{SWEET}_{\text{Pnx}}$). This setup achieves the highest $F_{0.5}$ score among text editing models, setting a new SOTA on QALB-2014 with 80.3. This improvement is statistically significant compared to Seq2Seq++ ($p < 0.05$) and is driven by a precision of 83.7. Its performance on ZAEBUC leads other edit tagging techniques but trails behind GPT-4o.

For our ensemble models (3-Ensemble), we combine the outputs of the top three non-LLM models per dataset. For QALB-2014 and ZAEBUC, this includes Seq2Seq++, SWEET^2, and the cascaded setup $\text{SWEET}^2_{\text{NoPnx}}$ + $\text{SWEET}_{\text{Pnx}}$. For MADAR CODA, we ensemble Seq2Seq++, SWEET, and the second-best SWEET model using CAMeLBERT-MSA (see Table 10 in Appendix C). The 3-Ensembles outperform single models, achieving SOTA results across all datasets, primarily through increased precision at the cost of recall. Adding GPT-4o's output to the ensemble further boosts performance (4-Ensemble), reaching an $F_{0.5}$ of 81.8 on QALB-2014, 86.9 on ZAEBUC, and 88.6 on MADAR CODA. These gains are statistically significant ($p < 0.05$) compared to the best baseline and the best non-ensemble model for each dataset.

Test Results Tables 5 and 6 present the Test results for MSA and DA GEC, using the best setups identified from the Dev sets. On QALB-2014, the cascaded setup $\text{SWEET}^2_{\text{NoPnx}}$ + $\text{SWEET}_{\text{Pnx}}$ sets a new SOTA with 80.5 $F_{0.5}$, outperforming the Seq2Seq and Seq2Seq++ baselines (statistically significant at $p < 0.05$). On QALB-2015, this setup matches Seq2Seq++ with an $F_{0.5}$ of 80.3. Similarly, on ZAEBUC, it achieves 83.1, on par with Seq2Seq++. On MADAR CODA, SWEET achieves 86.5, outperforming Seq2Seq++ (though not statistically significant). Our ensemble models further enhance performance across all datasets, reaching 81.7 on QALB-2014, 82.9 on QALB-2015, 87.2 on ZAEBUC, and 88.9 on MADAR CODA. Notably, adding GPT-4o's output to the ensemble (i.e., the 4-Ensemble) yields statistically significant improvements for MSA GEC. On MADAR CODA, the 3-Ensemble already achieves statistically significant gains, while the addition of GPT-4o does not lead to further improvement.

5.1 Error Analysis

Table 7 presents specific error type performance over the Dev sets of QALB-2014, ZAEBUC, and MADAR CODA. We conduct automatic error analysis using ARETA (Belkebir and Habash, 2021), an automatic MSA error type annotation tool. ARETA defines error types based on seven classes covering: orthography (**O**), morphology (**M**), syntax (**X**), semantics (**S**), punctuation (**P**), merges (**Merge-Beginning/Merge-Inside**), and splits (**Split**).

On QALB-2014, the cascaded setup $\text{SWEET}^2_{\text{NoPnx}}$ + $\text{SWEET}_{\text{Pnx}}$ outperforms the Seq2Seq++ baseline on most error types, achieving a macro $F_{0.5}$ of 71.6. The 4-Ensemble provides a modest improvement, reaching 71.9. On ZAEBUC, while the cascaded SWEET setup does not surpass Seq2Seq++ in overall performance (Table 3), it achieves higher scores on most individual error types, with a macro $F_{0.5}$ of 71.2. This stems from the skewed distribution of error types in the ZAEBUC Dev set, which is dominated by correct words (**C**) and frequent errors like **O** and **Merge**, categories where both models perform similarly. The error types where the cascaded SWEET model excels are relatively infrequent (see Table 16 in Appendix G). Notably, while the 4-Ensemble yields the best overall GEC results, it falls short of Seq2Seq++ and the cascaded SWEET model in error-type performance, likely due to prioritizing precision over recall (Table 3).

On MADAR CODA, neither SWEET nor the 4-Ensemble surpasses Seq2Seq++ in error-type performance. While SWEET performs slightly better in terms of DA GEC, the improvement is not statistically significant and is driven primarily by precision; in contrast, Seq2Seq++ claims the highest recall (Table 4). The 4-Ensemble further increases precision but at the cost of recall. The high proportion of the unknown (**UNK**) errors also highlights ARETA's limitations in capturing dialect-specific errors, as it was primarily designed for MSA.

5.2 Runtime Performance

Table 8 compares our text editing models to the Seq2Seq models from Alhafni et al. (2023) in terms of model size, initialization time, and inference runtime. Initialization and inference times were averaged over 10 runs on the QALB-2014 Dev set using a single A100 GPU with a batch size of 32. The reported values for Seq2Seq++ reflect the combined size, initialization, and inference times of all its components. Our SWEET model is 4x smaller than Seq2Seq++, while the cascaded system $\text{SWEET}^2_{\text{NoPnx}}$ + $\text{SWEET}_{\text{Pnx}}$ is about half the Seq2Seq++ model size. In terms of speed, SWEET initializes 19x faster than Seq2Seq++, while the cascaded system achieves a 9x initialization speedup. For inference, SWEET is also 19x faster, SWEET^2 is 9x faster, and the cascaded setup is 6x faster. Compared to the vanilla Seq2Seq model, SWEET runs 6x faster, SWEET^2 is 3x faster, and the cascaded system is twice as fast. Although the 3-Ensemble setup achieves the best performance, it is the largest in model size and the slowest overall.

	Params	Time	
		Init.	Run
A'2023 (Seq2Seq)	139M	1.7	70.7
A'2023 (Seq2Seq++)	502M	24.7	218.5
SWEET	135M	1.3	11.6
SWEET^2	135M	1.3	23.2
$\text{SWEET}^2_{\text{NoPnx}}$ + $\text{SWEET}_{\text{Pnx}}$	270M	2.7	34.8
3-Ensemble	908M	28.7	276.4

Table 8: Number of parameters (Params.), initialization time (Init.), and runtime for different models on the Dev set of QALB-2014. Init. and runtime are in seconds and averaged over 10 runs on a single A100 GPU using a batch size of 32.

6 Conclusion and Future Work

We introduced a data-driven text editing approach that eliminates the need for predefined language-specific edits. By applying it to Arabic, a diglossic and morphologically rich language, we studied the impact of different edit representations on model performance. Our models set new SOTA results on two Arabic GEC benchmarks and matched top-performing systems on two others. Moreover, they offer a significant efficiency advantage, running over six times faster than existing Arabic GEC systems, making them more suitable for practical deployment. We also explored how ensemble models contribute to further performance improvements.

In future work, we plan to extend this approach to other languages and dialectal varieties (Jarrar et al., 2016; Khalifa et al., 2018) and investigate its potential for generating synthetic data for GEC (Li et al., 2022; Zhang et al., 2022; Stahlberg and Kumar, 2024). We also plan to explore other ensembling approaches (Qorib and Ng, 2023; Qorib et al., 2022).

Limitations

While our work demonstrates promising results, there are several considerations that could impact its broader applicability. One limitation is the use of closed-source commercial LLMs, which introduces a degree of uncertainty, as these models may undergo undisclosed updates over time. Such changes could affect the reproducibility of our results. Additionally, we did not report on L2 Arabic GEC, which could provide valuable insights into how our approach generalizes to second-language learners errors. We also did not explore multilingual transformer encoders, as we hypothesize that monolingual models would be more effective for Arabic GEC. However, future work is needed to verify this assumption. Finally, our analysis focused on Arabic, which may limit the generalizability of our findings to languages with different error correction challenges.

Ethical Considerations

GEC systems can aid in identifying and correcting errors, but they also raise ethical concerns. Misidentifications or miscorrections may frustrate learners, and GEC tools should complement, not replace, human judgment. There is also the risk of malicious use, such as profiling learners based on error patterns, which could lead to bias or privacy issues. It is important to use these systems responsibly to protect end users.

Acknowledgments

We thank Ted Briscoe for helpful discussions and constructive feedback. We also acknowledge the support of the High Performance Computing Center at New York University Abu Dhabi.

References

Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. 2021. ARBERT & MARBERT: Deep bidirectional transformers for Arabic. In *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 7088–7105, Online. Association for Computational Linguistics.

Bashar Alhafni, Sarah Al-Towaity, Ziyad Fawzy, Fatema Nassar, Fadhl Eryani, Houda Bouamor, and Nizar Habash. 2024. Exploiting dialect identification in automatic dialectal text normalization. In *Proceedings of The Second Arabic Natural Language Processing Conference*, pages 42–54, Bangkok, Thailand. Association for Computational Linguistics.

Bashar Alhafni, Go Inoue, Christian Khairallah, and Nizar Habash. 2023. Advancements in Arabic grammatical error detection and correction: An empirical investigation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6430–6448, Singapore. Association for Computational Linguistics.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France. European Language Resource Association.

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4260–4270, Hong Kong, China. Association for Computational Linguistics.

Riadh Belkebir and Nizar Habash. 2021. Automatic error type annotation for Arabic. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 596–606, Online. Association for Computational Linguistics.

Houda Bouamor, Nizar Habash, Mohammad Salameh, Wajdi Zaghouani, Owen Rambow, Dana Abdulrahim, Ossama Obeid, Salam Khalifa, Fadhl Eryani, Alexander Erdmann, and Kemal Oflazer. 2018. The MADAR Arabic dialect corpus and lexicon. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Fethi Bougares and Houda Bouamor. 2015. UMMU@QALB-2015 shared task: Character and word level SMT pipeline for automatic error correction of Arabic text. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, pages 166–172, Beijing, China. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2023. Grammatical error correction: A survey of the state of the art. *Computational Linguistics*, pages 643–701.

Steven Coyne, Keisuke Sakaguchi, Diana Galvan-Sosa, Michael Zock, and Kentaro Inui. 2023. Analyzing

the performance of gpt-3.5 and gpt-4 in grammatical error correction. *Preprint*, arXiv:2303.14342.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

Christopher Davis, Andrew Caines, Øistein E. Andersen, Shiva Taslimipoor, Helen Yannakoudakis, Zheng Yuan, Christopher Bryant, Marek Rei, and Paula Buttery. 2024. Prompting open-source and commercial language models for grammatical error correction of English learner text. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11952–11967, Bangkok, Thailand. Association for Computational Linguistics.

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.

Mona Diab, Mohamed Al-Badrashiny, Maryam Aminian, Mohammed Attia, Heba Elfardy, Nizar Habash, Abdelati Hawwari, Wael Salloum, Pradeep Dasigi, and Ramy Eskander. 2014. Tharwa: A large scale dialectal Arabic - Standard Arabic - English lexicon. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3782–3789, Reykjavik, Iceland. European Language Resources Association (ELRA).

Fadhl Eryani, Nizar Habash, Houda Bouamor, and Salam Khalifa. 2020. A spelling correction corpus for multiple Arabic dialects. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4130–4138, Marseille, France. European Language Resources Association.

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh. 2013. Processing spontaneous orthography. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 585–595, Atlanta, Georgia. Association for Computational Linguistics.

Tao Fang, Shu Yang, Kaixin Lan, Derek F. Wong, Jinpeng Hu, Lidia S. Chao, and Yue Zhang. 2023. Is chatgpt a highly fluent grammatical error correction system? a comprehensive evaluation. *Preprint*, arXiv:2304.01746.

Noura Farra, Nadi Tomeh, Alla Rozovskaya, and Nizar Habash. 2014. Generalized character-level spelling error correction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–167, Baltimore, Maryland. Association for Computational Linguistics.

Charles F Ferguson. 1959. Diglossia. *Word*, 15(2):325–340.

Abbas Ghaddar, Yimeng Wu, Sunyam Bagga, Ahmad Rashid, Khalil Bibi, Mehdi Rezagholizadeh, Chao Xing, Yasheng Wang, Xinyu Duan, Zhefeng Wang, Baoxing Huai, Xin Jiang, Qun Liu, and Phillippe Langlais. 2022. Revisiting pre-trained language models and their evaluation for Arabic natural language processing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3135–3151, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, Florence, Italy. Association for Computational Linguistics.

Nizar Habash, Mona Diab, and Owen Rambow. 2012a. Conventional orthography for dialectal Arabic. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 711–718, Istanbul, Turkey. European Language Resources Association (ELRA).

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, Sara Hassan, Faisal Al-Shargi, Sakhar Alkhereyf, Basma Abdulkareem, Ramy Eskander, Mohammad Salameh, and Hind Saddiki. 2018. Unified guidelines and resources for Arabic dialect orthography. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012b. A morphological analyzer for Egyptian Arabic. In *Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology*, pages 1–9, Montréal, Canada. Association for Computational Linguistics.

Nizar Habash and David Palfreyman. 2022. ZAEBUC: An annotated Arabic-English bilingual writer corpus. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 79–88, Marseille, France. European Language Resources Association.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, pages 15–22. Springer, Netherlands.

Go Inoue, Bashar Alhafni, Nurpeiis Baimukan, Houda Bouamor, and Nizar Habash. 2021. The interplay of variant, size, and task type in Arabic pre-trained language models. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 92–104, Kyiv, Ukraine (Virtual). Association for Computational Linguistics.

Mustafa Jarrar, Nizar Habash, Faeq Alrimawi, Diyam Akra, and Nasser Zalmout. 2016. Curras: an anno-

tated corpus for the Palestinian Arabic dialect. *Language Resources and Evaluation*, pages 1–31.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606, New Orleans, Louisiana. Association for Computational Linguistics.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4248–4254, Online. Association for Computational Linguistics.

Masahiro Kaneko and Naoaki Okazaki. 2023. Reducing sequence length by predicting edit spans with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10017–10029, Singapore. Association for Computational Linguistics.

Masahiro Kaneko and Naoaki Okazaki. 2024. Controlled generation with prompt insertion for natural language explanations in grammatical error correction. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3955–3961, Torino, Italia. ELRA and ICCL.

Anisia Katinskaia and Roman Yangarber. 2024. GPT-3.5 for grammatical error correction. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 7831–7843, Torino, Italia. ELRA and ICCL.

Satoru Katsumata and Mamoru Komachi. 2020. Stronger baselines for grammatical error correction using a pretrained encoder-decoder model. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 827–832, Suzhou, China. Association for Computational Linguistics.

Salam Khalifa, Nizar Habash, Fadhl Eryani, Ossama Obeid, Dana Abdulrahim, and Meera Al Kaabi. 2018. A morphologically annotated corpus of emirati Arabic. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Salam Khalifa, Nasser Zalmout, and Nizar Habash. 2020. Morphological analysis and disambiguation for Gulf Arabic: The interplay between resources and methods. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3895–3904, Marseille, France. European Language Resources Association.

Sang Kwon, Gagan Bhatia, El Moatez Billah Nagoudi, and Muhammad Abdul-Mageed. 2023. Beyond English: Evaluating LLMs for Arabic grammatical error correction. In *Proceedings of ArabicNLP 2023*, pages 101–119, Singapore (Hybrid). Association for Computational Linguistics.

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Zuchao Li, Kevin Parnow, and Hai Zhao. 2022. Incorporating rich syntax information in grammatical error correction. *Information Processing & Management*, 59(3):102891.

Mengsay Loem, Masahiro Kaneko, Sho Takase, and Naoaki Okazaki. 2023. Exploring effectiveness of GPT-3 in grammatical error correction: A study on performance and controllability in prompt-based methods. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 205–219, Toronto, Canada. Association for Computational Linguistics.

Agnes Luhtaru, Elizaveta Korotkova, and Mark Fishel. 2024. No error left behind: Multilingual grammatical error correction with pre-trained translation models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1209–1222, St. Julian's, Malta. Association for Computational Linguistics.

Mohamed Maamouri, Ann Bies, Seth Kulick, Michael Ciul, Nizar Habash, and Ramy Eskander. 2014. Developing an egyptian arabic treebank: Impact of dialectal morphology on annotation and tool development. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Samar Mohamed Magdy, Fakhraddin Alwajih, Sang Yun Kwon, Reem Abdel-Salam, and Muhammad Abdul-Mageed. 2024. Gazelle: An instruction dataset for Arabic writing assistance. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 16027–16054, Miami, Florida, USA. Association for Computational Linguistics.

Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. EdiT5: Semi-autoregressive text editing with t5 warm-start. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2126–2138, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jonathan Mallinson, Aliaksei Severyn, Eric Malmi, and Guillermo Garrido. 2020. FELIX: Flexible text editing through tagging and insertion. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1244–1255, Online. Association for Computational Linguistics.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International*

Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Stuart Mesham, Christopher Bryant, Marek Rei, and Zheng Yuan. 2023. An extended sequence tagging vocabulary for grammatical error correction. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1608–1619, Dubrovnik, Croatia. Association for Computational Linguistics.

Masato Mita, Keisuke Sakaguchi, Masato Hagiwara, Tomoya Mizumoto, Jun Suzuki, and Kentaro Inui. 2024. Towards automated document revision: Grammatical error correction, fluency edits, and beyond. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, pages 251–265, Mexico City, Mexico. Association for Computational Linguistics.

Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. 2014. The first QALB shared task on automatic text correction for Arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 39–47, Doha, Qatar. Association for Computational Linguistics.

Michael Nawar. 2015. CUFE@QALB-2015 shared task: Arabic error correction system. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, pages 133–137, Beijing, China. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Ossama Obeid, Go Inoue, and Nizar Habash. 2022. Camelira: An Arabic multi-dialect morphological disambiguator. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 319–326, Abu Dhabi, UAE. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Andrii Liubonko, Oleksandr Skurzhanskyi, Artem Chernodub, Oleksandr Korniienko, and Igor Samokhin. 2024. Pillars of grammatical error correction: Comprehensive inspection of contemporary approaches in the era of large language models. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, pages 17–33, Mexico City, Mexico. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1094–1101, Reykjavik, Iceland. European Language Resources Association (ELRA).

Muhammad Reza Qorib, Seung-Hoon Na, and Hwee Tou Ng. 2022. Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1964–1974, Seattle, United States. Association for Computational Linguistics.

Muhammad Reza Qorib and Hwee Tou Ng. 2023. System combination via quality estimation for grammatical error correction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12746–12759, Singapore. Association for Computational Linguistics.

Vipul Raheja, Dimitris Alikaniotis, Vivek Kulkarni, Bashar Alhafni, and Dhruv Kumar. 2024. mEdIT: Multilingual text editing via instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 979–1001, Mexico City, Mexico. Association for Computational Linguistics.

Vipul Raheja, Dhruv Kumar, Ryan Koo, and Dongyeop Kang. 2023. CoEdIT: Text editing by task-specific instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5274–5291, Singapore. Association for Computational Linguistics.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online. Association for Computational Linguistics.

Alla Rozovskaya, Houda Bouamor, Nizar Habash, Wajdi Zaghouani, Ossama Obeid, and Behrang Mohit. 2015. The second QALB shared task on automatic text correction for Arabic. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, pages 26–35, Beijing, China. Association for Computational Linguistics.

Alla Rozovskaya, Nizar Habash, Ramy Eskander, Noura Farra, and Wael Salloum. 2014. The Columbia system in the QALB-2014 shared task on Arabic error correction. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 160–164, Doha, Qatar. Association for Computational Linguistics.

Neha Sengupta, Sunil Kumar Sahu, Bokang Jia, Satheesh Katipomu, Haonan Li, Fajri Koto, William Marshall, Gurpreet Gosal, Cynthia Liu, Zhiming Chen, Osama Mohammed Afzal, Samta Kamboj, Onkar Pandit, Rahul Pal, Lalit Pradhan, Zain Muhammad Mujahid, Massa Baali, Xudong Han, Sondos Mahmoud Bsharat, and 13 others. 2023. Jais and jais-chat: Arabic-centric foundation and instruction-tuned open generative large language models. *Preprint*, arXiv:2308.16149.

Aiman Solyman, Zhenyu Wang, Qian Tao, Arafat Abdulgader Mohammed Elhag, Rui Zhang, and Zeinab Mahmoud. 2022. Automatic Arabic grammatical error correction based on expectation-maximization routing and target-bidirectional agreement. *Knowledge-Based Systems*, 241:108180.

Aiman Solyman, Marco Zappatore, Wang Zhenyu, Zeinab Mahmoud, Ali Alfatemi, Ashraf Osman Ibrahim, and Lubna Abdelkareim Gabralla. 2023. Optimizing the impact of data augmentation for low-resource grammatical error correction. *Journal of King Saud University - Computer and Information Sciences*, 35(6):101572.

Aiman Solyman, Wang Zhenyu, Tao Qian, Arafat Abdulgader Mohammed Elhag, Muhammad Toseef, and Zeinab Aleibeid. 2021. Synthetic data with neural machine translation for automatic correction in Arabic grammar. *Egyptian Informatics Journal*, 22(3):303–315.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159, Online. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2024. Synthetic data generation for low-resource grammatical error correction with tagged corruption models. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, pages 11–16, Mexico City, Mexico. Association for Computational Linguistics.

Milan Straka, Jakub Náplava, and Jana Straková. 2021. Character transformations for non-autoregressive GEC tagging. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 417–422, Online. Association for Computational Linguistics.

Maksym Tarnavskyi, Artem Chernodub, and Kostiantyn Omelianchuk. 2022. Ensembling and knowledge distilling of large sequence taggers for grammatical error correction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3842–3852, Dublin, Ireland. Association for Computational Linguistics.

Fanar Team, Ummar Abbas, Mohammad Shahmeer Ahmad, Firoj Alam, Enes Altinisik, Ehsannedin Asgari, Yazan Boshmaf, Sabri Boughorbel, Sanjay Chawla, Shammur Chowdhury, Fahim Dalvi, Kareem Darwish, Nadir Durrani, Mohamed Elfeky, Ahmed Elmagarmid, Mohamed Eltabakh, Masoomali Fatehkia, Anastasios Fragkopoulos, Maram Hasanain, and 23 others. 2025. Fanar: An arabic-centric multimodal generative ai platform. *Preprint*, arXiv:2501.13944.

Zhaohong Wan, Xiaojun Wan, and Wenguang Wang. 2020. Improving grammatical error correction with data augmentation by editing latent representation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2202–2212, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Daniel Watson, Nasser Zalmout, and Nizar Habash. 2018. Utilizing character and word embeddings for text normalization with sequence-to-sequence models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 837–843, Brussels, Belgium. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Haoran Wu, Wenxuan Wang, Yuxuan Wan, Wenxiang Jiao, and Michael Lyu. 2023. Chatgpt or grammarly? evaluating chatgpt on grammatical error correction benchmark. *Preprint*, arXiv:2303.13648.

Zheng Yuan and Christopher Bryant. 2021. Document-level grammatical error correction. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 75–84, Online. Association for Computational Linguistics.

Zheng Yuan, Felix Stahlberg, Marek Rei, Bill Byrne, and Helen Yannakoudakis. 2019. Neural and FST-based approaches to grammatical error correction. In

Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 228–239, Florence, Italy. Association for Computational Linguistics.

Zheng Yuan, Shiva Taslimipoor, Christopher Davis, and Christopher Bryant. 2021. Multi-class grammatical error detection for correction: A tale of two systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8722–8736, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wajdi Zaghouani, Nizar Habash, Houda Bouamor, Alla Rozovskaya, Behrang Mohit, Abeer Heider, and Kemal Oflazer. 2015. Correction annotation for non-native Arabic texts: Guidelines and corpus. In *Proceedings of the 9th Linguistic Annotation Workshop*, pages 129–139, Denver, Colorado, USA. Association for Computational Linguistics.

Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Ossama Obeid, Nadi Tomeh, Alla Rozovskaya, Noura Farra, Sarah Alkuhlani, and Kemal Oflazer. 2014. Large scale Arabic error annotation: Guidelines and framework. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Nasser Zalmout, Alexander Erdmann, and Nizar Habash. 2018. Noise-robust morphological disambiguation for dialectal Arabic. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 953–964, New Orleans, Louisiana. Association for Computational Linguistics.

Nasser Zalmout and Nizar Habash. 2020. Joint diacritization, lemmatization, normalization, and fine-grained morphological tagging. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8297–8307, Online. Association for Computational Linguistics.

Yu Zhang, Yue Zhang, Leyang Cui, and Guohong Fu. 2023. Non-autoregressive text editing with copy-aware latent alignments. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7075–7085, Singapore. Association for Computational Linguistics.

Yue Zhang, Bo Zhang, Zhenghua Li, Zuyi Bao, Chen Li, and Min Zhang. 2022. SynGEC: Syntax-enhanced grammatical error correction with a tailored GEC-oriented parser. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2518–2531, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. 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 156–165, Minneapolis, Minnesota. Association for Computational Linguistics.

Houquan Zhou, Yumeng Liu, Zhenghua Li, Min Zhang, Bo Zhang, Chen Li, Ji Zhang, and Fei Huang. 2023. Improving Seq2Seq grammatical error correction via decoding interventions. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7393–7405, Singapore. Association for Computational Linguistics.

A Hyperparameters

We use Hugging Face's Transformers to build our edit taggers. Models trained on QALB-2014 or MADAR CODA are fine-tuned for 10 epochs using a learning rate of 5e-5, a batch size of 32, a maximum sequence length of 512, and a seed of 42 on a single A100 GPU. For models trained on QALB-2014 with the tenfold upsampled ZAEBUC, we use the same hyperparameters but run training for 15 epochs. At the end of fine-tuning, we pick the best checkpoint based on the performance on the Dev sets by using the M^2 scorer.

B Edit Coverage

				QALB-2014			ZAEBUC			MADAR CODA		
Input	**Comp.**	**Subset**	**Prune**	**Edits**	**OOV%**	**$F_{0.5}$**	**Edits**	**OOV%**	**$F_{0.5}$**	**Edits**	**OOV%**	**$F_{0.5}$**
Word	✗	All	-	16,221	1.00%	98.4	1,097	2.94%	96.2	1,228	1.52%	98.0
Subword	✗	All	-	9,060	0.36%	98.7	905	1.85%	96.5	677	0.55%	98.1
Word	✓	All	-	10,410	1.00%	98.4	687	2.94%	96.2	741	1.52%	98.0
Subword	✓	All	-	6,170	0.36%	98.7	563	1.85%	96.5	454	0.55%	98.1
Subword	✓	NoPnx	-	4,799	0.27%	98.8	498	1.74%	96.2	-	-	-
Subword	✓	Pnx	-	160	0.01%	99.4	23	0.06%	99.9	-	-	-
Subword	✓	All	10	683	0.75%	98.1	58	3.71%	93.9	84	1.33%	96.2
Subword	✓	All	20	442	1.02%	97.7	35	4.67%	92.6	52	2.02%	94.1
Subword	✓	All	30	329	1.24%	97.4	27	5.26%	91.8	45	2.28%	93.4
Subword	✓	NoPnx	10	520	0.56%	98.2	52	3.39%	93.7	-	-	-
Subword	✓	NoPnx	20	335	0.75%	97.8	30	4.31%	92.3	-	-	-
Subword	✓	NoPnx	30	250	0.92%	97.5	22	4.90%	91.4	-	-	-
Subword	✓	Pnx	10	48	0.02%	99.4	6	0.11%	99.9	-	-	-
Subword	✓	Pnx	20	35	0.05%	99.4	6	0.11%	99.9	-	-	-
Subword	✓	Pnx	30	29	0.05%	99.3	6	0.11%	99.9	-	-	-

Table 9: Edit statistics on QALB-2014, ZAEBUC and MADAR CODA. **Input** is the input unit (word or subword). **Comp.** indicates whether the edit is compressed. **Subset** specifies whether the edits capture all errors, punctuation-only errors (Pnx), or non-punctuation errors (NoPnx). **Edits** represents the total number of unique edits in the training set of each dataset. **OOV%** is the percentage of out-of-vocabulary edits (non-unique) in the Dev set of each dataset.

C Edit Tagging Results

					QALB-2014				ZAEBUC				MADAR CODA			
Model	**Input**	**Comp.**	**Subset**	**Prune**	**P**	**R**	**F_1**	**$F_{0.5}$**	**P**	**R**	**F_1**	**$F_{0.5}$**	**P**	**R**	**F_1**	**$F_{0.5}$**
AraBERTv02	Word	✗	All	-	81.0	64.3	71.7	77.0	84.8	69.5	76.4	81.2	87.9	66.5	75.7	82.6
AraBERTv02	Subword	✗	All	-	81.0	67.8	73.8	77.9	84.4	71.3	77.3	81.4	87.6	76.8	81.9	85.2
AraBERTv02	Word	✓	All	-	80.8	66.6	73.0	77.5	83.8	71.4	77.1	81.0	85.6	76.9	81.0	83.7
AraBERTv02	Subword	✓	All	-	81.1	**69.1**	74.6	78.4	84.3	**72.9**	78.2	81.7	86.9	**79.2**	**82.9**	85.2
ARBERTv2	Word	✗	All	-	78.9	57.7	66.7	73.5	82.2	54.8	65.8	74.8	86.4	61.0	71.5	79.8
ARBERTv2	Subword	✗	All	-	78.7	60.8	68.6	74.3	79.7	58.1	67.2	74.2	84.5	69.0	76.0	80.8
ARBERTv2	Word	✓	All	-	77.8	61.4	68.6	73.8	80.7	62.8	70.6	76.3	81.8	68.4	74.5	78.7
ARBERTv2	Subword	✓	All	-	78.6	60.0	68.0	74.0	82.7	62.1	70.9	77.5	84.2	70.8	77.0	81.2
CAMeLBERT	Word	✗	All	-	81.2	61.5	70.0	76.3	84.6	66.4	74.4	80.2	88.3	66.4	75.8	82.8
CAMeLBERT	Subword	✗	All	-	80.4	65.2	72.0	76.9	83.5	69.3	75.8	80.2	87.1	76.8	81.6	84.8
CAMeLBERT	Word	✓	All	-	79.9	65.4	71.9	76.5	84.2	69.3	76.0	80.7	85.6	76.0	80.6	83.5
CAMeLBERT	Subword	✓	All	-	80.6	67.4	73.4	77.6	84.6	70.8	77.1	81.4	87.0	78.8	82.7	85.2
AraBERTv02	Subword	✓	All	10	**81.8**	68.8	**74.7**	**78.8**	84.5	71.9	77.7	81.6	**89.1**	75.5	81.7	**86.0**
AraBERTv02	Subword	✓	All	20	81.4	68.6	74.4	78.5	85.3	72.0	78.1	82.2	87.7	73.1	79.8	84.4
AraBERTv02	Subword	✓	All	30	81.6	68.1	74.3	78.5	**85.8**	72.3	**78.4**	**82.7**	88.3	72.1	79.4	84.5
CAMeLBERT	Subword	✓	All	10	81.2	67.4	73.7	78.0	85.1	71.0	77.4	81.8	88.4	76.3	81.9	85.7
CAMeLBERT	Subword	✓	All	20	81.3	66.7	73.3	77.9	84.4	70.1	76.6	81.1	88.2	72.6	79.6	84.6
CAMeLBERT	Subword	✓	All	30	81.1	67.5	73.7	77.9	84.7	70.0	76.6	81.3	88.7	71.3	79.1	84.6
AraBERTv02	Subword	✓	NoPnx	-	88.3	77.7	82.6	85.9	87.2	77.0	81.8	85.0	-	-	-	-
AraBERTv02	Subword	✓	NoPnx	10	88.8	78.1	83.1	86.4	87.6	76.1	81.4	85.0	-	-	-	-
AraBERTv02	Subword	✓	NoPnx	20	89.0	77.8	83.0	86.5	87.9	75.8	81.4	85.1	-	-	-	-
AraBERTv02	Subword	✓	NoPnx	30	89.4	77.5	83.0	86.7	88.1	76.8	82.1	85.6	-	-	-	-
AraBERTv02	Subword	✓	Pnx	-	90.6	83.0	86.6	89.0	96.8	94.0	95.4	96.2	-	-	-	-
AraBERTv02	Subword	✓	Pnx	10	89.5	83.6	86.5	88.3	96.9	93.8	95.3	96.3	-	-	-	-
AraBERTv02	Subword	✓	Pnx	20	90.7	82.8	86.5	89.0	96.7	93.6	95.1	96.1	-	-	-	-
AraBERTv02	Subword	✓	Pnx	30	90.1	83.6	86.7	88.7	96.5	94.0	95.2	96.0	-	-	-	-

Table 10: MSA and DA GEC results on the Dev sets of QALB-2014, ZAEBUC, and MADAR CODA. **Input** is the input unit (word or subword). **Comp.** indicates whether the edit is compressed. **Subset** specifies whether the edits capture all errors, punctuation-only errors (Pnx), or non-punctuation errors (NoPnx). NoPnx models are evaluated after removing punctuation, while Pnx models are evaluated on a version of the Dev set where all non-punctuation errors are corrected. Pruning experiments were conducted using the top two models (AraBERTv02 and CAMeLBERT), while punctuation segregation experiments used the best model (AraBERTv02). Best **All** results are in bold; best **NoPnx** and **Pnx** results are underlined.

D LLMs Results

Model	P-Lang	Shots	QALB-2014 P	R	F_1	$F_{0.5}$	ZAEBUC P	R	F_1	$F_{0.5}$	MADAR CODA P	R	F_1	$F_{0.5}$	Avg. $F_{0.5}$
GPT-3.5-turbo	EN	0	70.6	54.8	61.7	66.7	70.8	70.3	70.5	70.7	22.8	17.7	19.9	21.5	53.0
GPT-3.5-turbo	EN	5	68.6	58.6	63.2	66.3	71.0	63.5	67.1	69.4	35.5	29.7	32.3	34.1	56.6
GPT-3.5-turbo	AR	0	70.0	58.5	63.7	67.3	68.3	71.3	69.8	68.9	24.2	22.7	23.4	23.9	53.3
GPT-3.5-turbo	AR	5	68.1	58.0	62.6	65.8	71.4	63.7	67.3	69.7	27.0	26.5	26.7	26.9	54.1
GPT-4o	EN	0	**82.1**	56.4	66.8	75.2	80.2	75.5	77.8	79.2	28.8	25.5	27.0	28.1	60.8
GPT-4o	EN	5	80.7	65.7	72.4	**77.2**	**86.5**	76.8	**81.3**	**84.3**	**53.7**	**54.4**	**54.1**	**53.8**	**71.8**
GPT-4o	AR	0	78.9	62.8	69.9	75.1	77.4	**77.7**	77.5	77.4	36.4	33.5	34.9	35.8	62.8
GPT-4o	AR	5	79.5	**66.8**	**72.6**	76.6	82.6	75.7	79.0	81.1	50.1	48.6	49.4	49.8	69.2
Fanar	EN	0	57.4	31.4	40.6	49.2	58.4	18.6	28.2	40.9	13.7	14.6	14.1	13.9	34.7
Fanar	EN	5	63.3	58.8	61.0	62.4	69.2	63.5	66.2	68.0	22.4	26.8	24.4	23.1	51.2
Fanar	AR	0	62.4	57.3	59.7	61.3	57.5	33.9	42.6	50.4	17.2	19.0	18.1	17.5	43.1
Fanar	AR	5	69.7	63.7	66.6	68.4	76.3	73.6	74.9	75.8	24.5	28.8	26.4	25.2	56.5
Jais-13B-Chat	EN	0	49.1	37.0	42.2	46.1	53.3	5.5	10.0	19.5	8.6	8.3	8.4	8.5	24.7
Jais-13B-Chat	EN	5	48.9	36.0	41.5	45.7	46.6	4.7	8.6	16.9	14.9	16.3	15.6	15.2	25.9
Jais-13B-Chat	AR	0	48.2	36.2	41.4	45.2	40.8	5.4	9.6	17.7	10.7	10.6	10.7	10.7	24.5
Jais-13B-Chat	AR	5	49.1	36.9	42.1	46.0	50.2	19.7	28.3	38.4	14.1	15.0	14.5	14.3	32.9

Table 11: LLMs results on MSA and DA GEC on the Dev sets of QALB-2014, ZAEBUC, and MADAR CODA. P-Lang is the prompt language either in English (EN) or Arabic (AR). Best average $F_{0.5}$ results for each LLM are underlined; best overall results are in bold.

E MSA GEC Results

	QALB-2014 P	R	F_1	$F_{0.5}$	ZAEBUC P	R	F_1	$F_{0.5}$
AraBART	83.2	64.9	72.9	78.7	87.3	70.6	78.1	83.4
AraT5+Morph+GED[43]	83.1	67.9	74.7	79.6	85.2	71.2	77.6	82.0
AraBART+Morph+GED[13]	83.9	65.7	73.7	79.5	87.6	73.9	80.2	84.5
GPT-3.5-turbo	68.6	58.6	63.2	66.3	71.0	63.5	67.1	69.4
GPT-4o	80.7	65.7	72.4	77.2	86.5	**76.8**	**81.3**	84.3
Fanar	69.7	63.7	66.6	68.4	76.3	73.6	74.9	75.8
Jais-13B-Chat	49.1	36.9	42.1	46.0	50.2	19.7	28.3	38.4
SWEET	81.8	68.8	74.7	78.8	85.8	72.3	78.4	82.7
$SWEET^2$	81.9	**70.4**	75.7	79.3	85.8	73.3	79.1	83.0
$SWEET^2_{NoPnx}$ + $SWEET_{Pnx}$	83.7	68.8	75.6	80.3†	86.7	73.9	79.8	83.8
3-Ensemble	84.9	68.8	**76.0**	81.1	89.6	72.8	80.3	85.6
4-Ensemble	**89.1**	61.6	72.8	**81.8**‡	**93.3**	68.3	78.9	**86.9**†

Table 12: MSA GEC results on the Dev sets of QALB-2014 and ZAEBUC. Best non-ensemble results are underlined; best overall results are in bold. † denotes statistical significance over the best baseline; ‡ denotes statistical significance over both the best baseline and the best non-ensemble model.

	QALB-2014				QALB-2015				ZAEBUC			
	P	**R**	**F_1**	**$F_{0.5}$**	**P**	**R**	**F_1**	**$F_{0.5}$**	**P**	**R**	**F_1**	**$F_{0.5}$**
AraBART	84.0	64.7	73.1	79.3	82.0	71.7	76.5	79.7	86.0	71.6	78.2	82.7
AraBART+GED[43]	84.2	65.4	73.6	79.6	81.2	72.4	76.5	79.3	85.4	72.6	78.5	82.5
AraBART+Morph+GED[43]	83.9	65.7	73.7	79.5	82.6	72.1	77.0	80.3	85.4	73.7	79.1	82.7
AraBART+GED[13]	84.1	65.0	73.3	79.4	81.5	72.7	76.8	79.5	85.9	73.4	79.2	83.1
GPT-4o	81.5	65.5	72.6	77.7	81.1	**74.3**	77.5	79.6	84.4	**75.9**	79.9	82.5
SWEET[2]	82.6	**69.5**	**75.5**	79.6	80.0	**74.3**	77.0	78.8	85.5	74.4	79.6	83.0
$\text{SWEET}^2_{\text{NoPnx}}$ + $\text{SWEET}_{\text{Pnx}}$	84.5	67.7	75.2	80.5†	82.2	73.6	77.7	80.3	85.7	74.1	79.5	83.1
3-Ensemble	85.7	67.4	75.4	81.3	83.7	73.3	**78.1**	81.3	89.7	73.7	**80.9**	85.9
4-Ensemble	**89.7**	60.2	72.0	**81.7**‡	**88.3**	66.7	76.0	**82.9**‡	**93.4**	68.9	79.3	**87.2**‡

Table 13: MSA GEC results on the Test sets of QALB-2014, QALB-2015 (L1), and ZAEBUC. Best non-ensemble results are underlined; best overall results are in bold. † denotes statistical significance over the best baseline; ‡ denotes statistical significance over both the best baseline and the best non-ensemble model.

F DA GEC Results

	P	**R**	**F_1**	**$F_{0.5}$**
AraT5	86.8	77.4	81.8	84.7
AraT5+City	87.6	**79.3**	83.3	85.8
GPT-3.5-turbo	35.5	29.7	32.3	34.1
GPT-4o	53.7	54.4	54.1	53.8
Fanar	24.5	28.8	26.4	25.2
Jais-13B-Chat	14.1	15.0	14.5	14.3
SWEET	89.1	75.5	81.7	86.0
SWEET[2]	87.5	73.5	79.9	84.3
3-Ensemble	91.7	77.4	**83.9**	88.4
4-Ensemble	**93.8**	72.5	81.8	**88.6**‡

Table 14: DA GEC results on the MADAR CODA Dev set. Best non-ensemble results are underlined; best overall results are in bold. ‡ denotes statistical significance over both the best baseline and the best non-ensemble model.

	P	**R**	**F_1**	**$F_{0.5}$**
AraT5	87.3	78.0	82.4	85.2
AraT5+DA Phrase	88.4	**79.0**	83.4	86.3
GPT-4o	56.1	54.8	55.5	55.9
SWEET	89.4	76.6	82.5	86.5
3-Ensemble	92.2	77.7	**84.3**	**88.9**
4-Ensemble	**94.0**	72.9	82.1	88.8‡

Table 15: DA GEC results on the Test set of MADAR CODA. Best non-ensemble results are underlined; best overall results are in bold. ‡ denotes statistical significance over both the best baseline and the best non-ensemble model.

G Error Type Statistics

	QALB-2014			ZAEBUC			MADAR CODA		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**	**Train**	**Dev**	**Test**
Delete	6,442	346	540	305	64	66	35	0	1
Merge-B	15,063	797	795	849	180	133	404	102	95
Merge-I	15,296	812	807	851	180	133	429	109	103
M	1,466	69	63	137	32	28	159	42	56
M+O	243	17	15	7	1	8	0	0	0
O	141,752	7,380	6,976	3,203	695	792	5,604	1,179	1,193
O+X	323	24	18	20	0	4	0	0	0
P	11,379	598	687	237	51	36	18	10	9
S	5,436	247	252	169	36	51	408	77	64
X	13,592	809	743	528	110	113	911	166	151
Split	7,828	432	399	49	10	10	279	58	69
UNK	6,835	331	300	361	78	61	2,235	430	649
C	795,510	41,875	39,690	18,411	3,839	3,683	29,285	6,627	6,456
	1,021,165	53,737	51,285	25,127	5,276	5,118	39,767	8,800	8,846

Table 16: Distribution of error types in QALB-2014, ZAEBUC, and MADAR CODA. UNK refers to unknown error types; C refers to correct words.

H Prompts

Task	P-Lang	Prompt
MSA GEC	EN	You are an Arabic grammatical error correction tool that can identify and correct grammatical and spelling errors in written text. Please identify and correct any grammatical and spelling errors in the following sentence marked with the tag <input> Input Sentence </input>. Make the minimal changes necessary to correct the sentence. Do not rephrase any parts of the sentence that are already grammatically correct, and avoid altering the meaning by adding or removing information. After making the corrections, output the revised sentence directly without providing any explanations. Remember to format the corrected output with the tag <output> Your Corrected Version </output>. Please start: <input> Input Sentence </input>
	AR	أنت أداة لتصحيح الأخطاء النحوية والإملائية في اللغة العربية، حيث يمكنك تحديد وتصحيح الأخطاء النحوية والإملائية في النصوص المكتوبة. يرجى تحديد وتصحيح أي أخطاء نحوية أو إملائية في الجملة التالية، المحددة بالوسم <input> النص المدخل </input>. قم بإجراء الحد الأدنى من التعديلات اللازمة لتصحيح الجملة. لا تعد صياغة أي أجزاء صحيحة نحويا، وتجنب تغيير المعنى من خلال إضافة أو حذف أي معلومات. بعد إجراء التصحيحات، قم بإخراج الجملة المصححة مباشرة دون أي تفسيرات. تذكر تنسيق النص المصحح باستخدام الوسم <output> النص المصحح </output> الرجاء البدء: <input> النص المدخل </input>
DA GEC	EN	You are a dialectal Arabic text normalization tool that can normalize dialectal Arabic text into the Conventional Orthography for Dialectal Arabic (CODA). CODA provides a standardized system for writing Arabic dialects, which are often written informally or phonetically. By using CODA, your task is to convert these informal, dialectal texts into a consistent, standardized orthographic form, making them more uniform while retaining the nuances of the original dialect. Please standardize the following sentence marked with the tag <input> Input Sentence </input> into the CODA convention. Avoid altering the meaning by adding or removing information. Make sure the normalized output sentence is in Arabic script. Output the normalized sentence directly without providing any explanations. Remember to format the CODA standardized output with the tag <output> Your CODA Version </output>. Please start: <input> Input Sentence </input>
	AR	أنت أداة لتصحيح النصوص العربية العامية، حيث يمكنك تصحيح النصوص العامية إلى الإملاء القياسي للعامية العربية (CODA). يوفر CODA نظاما موحدا لكتابة اللهجات العربية التي تكتب غالبا بطريقة غير رسمية أو صوتية. باستخدام CODA، مهمتك هي تصحيح هذه النصوص العامية غير الرسمية إلى شكل إملائي موحد ومتسق، مع الحفاظ على الفروق الدقيقة للهجة الأصلية. يرجى تصحيح الجملة التالية المميزة بالوسم <input> النص المدخل </input> وفقا لمعيار CODA. تجنب تغيير المعنى عن طريق إضافة أو إزالة معلومات. أخرج الجملة مباشرة دون أي تفسيرات. تذكر تنسيق النص المصحح باستخدام الوسم <output> النص المصحح </output> الرجاء البدء: <input> النص المدخل </input>

Figure 2: 0-shot prompts used to evaluate LLMs performance on MSA and DA GEC. P-Lang is the prompt language either in English (EN) or Arabic (AR).

Task	P-Lang	Prompt
MSA GEC	EN	You are an Arabic grammatical error correction tool that can identify and correct grammatical and spelling errors in written text. We will provide you with example sentences marked with the tag <input> SRC </input>, which contain grammatical and spelling errors. These sentences are followed by the corrected versions, marked with <output> TGT </output>, as reviewed and edited by human experts. Please identify and correct any grammatical and spelling errors in the following sentence marked with the tag <input> SRC </input>. Make the minimal changes necessary to correct the sentence. Do not rephrase any parts of the sentence that are already grammatically correct, and avoid altering the meaning by adding or removing information. After making the corrections, output the revised sentence directly without providing any explanations. Here are some in-context examples: (1) <input> Input Sentence </input>: <output> Corrected Sentence </output> (2) <input> Input Sentence </input>: <output> Corrected Sentence </output> (3) <input> Input Sentence </input>: <output> Corrected Sentence </output> (4) <input> Input Sentence </input>: <output> Corrected Sentence </output> (5) <input> Input Sentence </input>: <output> Corrected Sentence </output> Please feel free to refer to these examples. Remember to format the corrected output with the tag <output> Your Corrected Version </output>. Please start: <input> Input Sentence </input>
	AR	أنت أداة لتصحيح الأخطاء النحوية والإملائية في اللغة العربية، حيث يمكنك تحديد وتصحيح الأخطاء النحوية والإملائية في النصوص المكتوبة .سنزودك بجمل تحتوي على أخطاء نحوية وإملائية، محددة بالوسم <input> النص المدخل </input>. تلي هذه الجمل النسخ المصححة، المحددة بالوسم <output> النص المصحح </output>، والتي تمت مراجعتها وتحريرها من قبل خبراء لغويين. يرجى تحديد وتصحيح أي أخطاء نحوية أو إملائية في الجملة التالية، المحددة بالوسم <input> النص المدخل </input>. قم بإجراء الحد الأدنى من التعديلات اللازمة لتصحيح الجملة. لا تعد صياغة أي أجزاء صحيحة نحويا، وتجنب تغيير المعنى من خلال إضافة أو حذف أي معلومات. بعد إجراء التصحيحات، قم بإخراج الجملة المصححة مباشرة دون أي تفسيرات. إليك بعض الأمثلة ضمن السياق: (١) <input> النص المدخل </input>: <output> النص المصحح </output> (٢) <input> النص المدخل </input>: <output> النص المصحح </output> (٣) <input> النص المدخل </input>: <output> النص المصحح </output> (٤) <input> النص المدخل </input>: <output> النص المصحح </output> (٥) <input> النص المدخل </input>: <output> النص المصحح </output> لا تتردد في الرجوع إلى هذه الأمثلة. تذكر تنسيق النص المصحح باستخدام الوسم <output> النص المصحح </output> الرجاء البدء: <input> النص المدخل </input>
DA GEC	EN	You are a dialectal Arabic text normalization tool that can normalize dialectal Arabic text into the Conventional Orthography for Dialectal Arabic (CODA). CODA provides a standardized system for writing Arabic dialects, which are often written informally or phonetically. By using CODA, your task is to convert these informal, dialectal texts into a consistent, standardized orthographic form, making them more uniform while retaining the nuances of the original dialect. We will provide you with example sentences marked with the tag <input> Input Sentence </input>, which are written in dialectal Arabic. These sentences are followed by their CODA versions, marked with <output> Corrected Sentence </output>, as reviewed and edited by human experts. Please standardize the following sentence marked with the tag <input> Input Sentence </input> into the CODA convention. Avoid altering the meaning by adding or removing information. Output the normalized sentence directly without providing any explanations. Here are some in-context examples: (1) <input> Input Sentence </input>: <output> Corrected Sentence </output> (2) <input> Input Sentence </input>: <output> Corrected Sentence </output> (3) <input> Input Sentence </input>: <output> Corrected Sentence </output> (4) <input> Input Sentence </input>: <output> Corrected Sentence </output> (5) <input> Input Sentence </input>: <output> Corrected Sentence </output> Please feel free to refer to these examples. Remember to format the corrected output with the tag <output> Your Corrected Version </output>. Please start: <input> Input Sentence </input>
	AR	أنت أداة لتصحيح النصوص العربية العامية، حيث يمكنك تصحيح النصوص العامية إلى الإملاء القياسي للعامية العربية (CODA). يوفر CODA نظاما موحدا لكتابة اللهجات العربية التي تكتب غالبا بطريقة غير رسمية أو صوتية. باستخدام CODA، مهمتك هي تصحيح هذه النصوص العامية غير الرسمية إلى شكل إملائي موحد ومتسق، مع الحفاظ على الفروق الدقيقة للهجة الأصلية. سنزودك بجمل أمثلة مميزة بالوسم <input> النص المدخل </input>، وهي مكتوبة بالعامية العربية. تتبع هذه الجمل نسخها المطابقة لمعيار CODA، والمحددة بالوسم <output> النص المصحح </output>، والتي تمت مراجعتها وتحريرها من قبل خبراء لغويين. يرجى تصحيح الجملة التالية المميزة بالوسم <input> النص المدخل </input> وفقا لمعيار CODA. تجنب تغيير المعنى عن طريق إضافة أو إزالة معلومات. أخرج الجملة مباشرة دون أي تفسيرات. إليك بعض الأمثلة ضمن السياق: (١) <input> النص المدخل </input>: <output> النص المصحح </output> (٢) <input> النص المدخل </input>: <output> النص المصحح </output> (٣) <input> النص المدخل </input>: <output> النص المصحح </output> (٤) <input> النص المدخل </input>: <output> النص المصحح </output> (٥) <input> النص المدخل </input>: <output> النص المصحح </output> لا تتردد في الرجوع إلى هذه الأمثلة. تذكر تنسيق النص المصحح باستخدام الوسم <output> النص المصحح </output> الرجاء البدء: <input> النص المدخل </input>

Figure 3: 5-shot prompts used to evaluate LLMs performance on MSA and DA GEC. P-Lang is the prompt language either in English (EN) or Arabic (AR).