Hybrid LLM and Rule-Based Synthetic Data Generation for Arabic Grammatical Error Correction

Mohamed Abdelrehim
Computer and Systems Engineering
Alexandria University
Alexandria, Egypt
mohamed.el.ms1@alexu.edu.eg

Marwan Torki
Computer and Systems Engineering
Alexandria University
Alexandria, Egypt
mtorki@alexu.edu.eg

Nagwa El-Makky
Computer and Systems Engineering
Alexandria University
Alexandria, Egypt
nagwamakky@alexu.edu.eg

***Abstract*—The correction of Arabic grammatical errors has recently gained interest due to the emergence of strong transformer models trained on internet-scale Arabic data and large language models capable of generalizing well to new tasks and languages. However, Arabic grammatical error correction still suffers from data scarcity, and the available datasets are insufficient to train efficient transformer models capable of performing well on all types of Arabic grammatical errors. In this work, we introduce a scalable recipe to generate synthetic data for Arabic grammatical error correction and control the distribution of error types with high granularity to boost performance in underrepresented error types in the original training data. Models trained on our synthesized data demonstrate enhanced performance in addressing challenging Arabic grammatical errors compared to those trained without it, and they rival state-of-the-art Arabic grammatical error correction systems.**

***Index Terms*—Natural Language Processing, Grammatical Error Correction, Large Language Models, Synthetic Data Generation**

I. Introduction

Automatic grammatical error correction (GEC) has many practical applications. It can enhance the language learning experience and can be used by many products, such as document editors and smartphone keyboards, to facilitate the user typing experience. Arabic GEC has many challenges that are common to many Arabic natural language processing tasks [1], [2]. Arabic is known for its rich morphological structure. Arabic words come in many forms and can be prefixed or suffixed in different ways depending on the grammatical context of a sentence. Arabic also has many variations in dialect. Due to these properties, Arabic GEC has been a challenging task.

In addition, the Arabic GEC suffers from resource scarcity. For example, the only adequately annotated dataset to train Arabic GEC has been the one released for the QALB-14 and QALB-15 shared tasks [3], [4] for a long time, which only contains around $20k$ training samples. The QALB-14 dataset is limited to the news commentary domain of a specific news website. The QALB-15 dataset is the same as QALB-14 but has been extended by nearly 500 samples from the domain of non-native speakers writing essay assignments. More recently, there have been efforts to build high-quality datasets for Arabic GEC [5], [6]. However, the available Arabic GEC datasets annotated by humans are too small to train neural models effectively. As a result, models trained on these datasets tend to perform poorly on out-of-domain testsets.

The generation of synthesized data has enabled state-of-the-art results for English GEC. Given the recent advancement of large language models (LLMs) trained on internet-scale data and finetuned for instruction following, synthesizing data using LLMs has emerged as a paradigm to mitigate the problem of unavailable or low-resource data [7].

In this paper, we explore how to generate synthetic data from large publicly available datasets to complement the available training data for Arabic GEC. We argue that models trained using state-of-the-art techniques in Arabic GEC using only QALB data may be prone to overfitting and performing poorly in challenging Arabic error types. Our key contributions are as follows.

- We introduce a recipe for generating synthesized Arabic GEC data by leveraging a prior distribution of target error types and integrating a rule-based approach with LLMs to introduce GEC errors.
- We develop a new synthetic dataset for Arabic GEC. We make the dataset and the code publicly available. [1]
- We demonstrate that models fine-tuned using real Arabic GEC data and our synthetic data show improved performance on challenging Arabic grammatical error types and compare favorably with recent state-of-the-art Arabic GEC systems.

The remainder of this paper is structured as follows: First, we review related work on GEC data generation and Arabic GEC. Next, we present our recipe for synthetic data generation. We then conduct experiments to evaluate the impact of training with our synthesized data. Finally, we conclude with remarks on our findings and potential directions for future research.

II. Related Work

Synthetic data generation is a widely recognized technique for achieving state-of-the-art results in English. Also, synthesizing errors in correct sentences makes it possible to use large-scale corpora that are not tied to a particular domain.

[1]https://github.com/mabdelrehim/hybrid-llm-rule-based-synthetic-data-generation-arabic-gec

Tagged corruption of grammatically correct sentences [8] is one of the synthetic data generation techniques that has shown good results. The main idea of the approach is to tag each word in the correct sentence with an error tag based on an established grammatical error taxonomy and then use those tags to corrupt each word. Then, a GEC model is trained using the incorrect sentences generated as input and the correct sentences as output. Stahlberg and Kumar [8] show that they can get improved results on testsets generated by native and non-native speakers when the synthetic dataset is optimized to match the distribution of development sets generated by each type of speakers respectively.

One key component for tagged corruption of grammatically correct sentences is a tool that automatically classifies grammatical errors according to some predefined taxonomy. For Arabic, ARETA [9] is a tool that has been developed for this purpose. It classifies Arabic grammatical errors according to the Arabic Learner Corpus (ALC) taxonomy established by Alfaifi and Atwell [10]. It also provides error tag-based metrics calculation which is useful for evaluating the performance of systems based on different grammatical error types.

Recently, Alhafni et al. [11] used pretrained sequence to sequence transformers such as AraBART [12] and AraT5 [13] to train GEC models for Arabic. They also use the ARETA tool to train grammatical error detection models and show improvements when a grammatical error detection model is used first to detect errors and the error information is incorporated into the GEC model.

Kwon et al. [14] compare between training sequence to sequence models, sequence-to-edits models [15], and prompting/finetuning LLMs for Arabic GEC. They find that sequence-to-sequence models pretrained on Arabic perform best on Arabic GEC. Mahmoud et al. [16] explore combing different edits generated from models trained with different seeds iteratively to create an ensemble.

Kwon et al. [14] also explore synthetic data creation approaches either through using a large language model to corrupt input sentences or using a reverse noising model trained to output corrupted sentences from correct sentences. They also use the Arabic Learner Corpus taxonomy [10] to guide large language models to generate sentences containing specific error types. Our approach is different in that we generate errors at the word level using a rule-based approach for some error classes and a large language model for other error classes. Our approach is more similar to that of Stahlberg and Kumar [8] in that we generate errors at the word level. This enables finer control over the distribution of error classes in the generated dataset and simplifies the task of synthesizing errors for the LLM.

An effort to create a dataset that can be used for Arabic writing assistance is the Gazelle dataset proposed by Magdy et al. [6]. They generate errors synthetically on the Arabic Tree Bank dataset [17]. For each error class, they have a set of predefined rules on how to generate that error. For example, for morphological errors, they have a set of word templates that are then used to reinflect a word to introduce each error. We rely on a similar method to generate errors for error classes that have clear rules. However, for more complex error classes like morphological errors or errors that require re-inflection such as mismatched gender error or mismatched plural, dual, or singular form, we rely on a large language model to generate these errors given a few shot prompt. Also, the Gazelle corpus is more intended for instruction finetuning of large language models.

Tibyan corpus [18] is also a synthetic corpus recently generated using a large language model by giving the model a seed example from which the model is prompted to generate a pair of correct - incorrect grammatical error correction sample. However, it contains only 49% of the Arabic grammatical error types in the ALC taxonomy.

III. Synthetic Data Generation

In this section, we detail how we synthetically generate training data to complement the available datasets for Arabic GEC. We first start by annotating the available training data, namely, the QALB-15 shared task data and the ZAEBUC data using the ARETA tool to measure the natural frequency of each error type.

The error class distribution in the training data is heavily skewed towards some errors and has a long tail. We include details on each error class along with its percentage in the training data in table III in the appendix. The most common errors are unnecessary punctuation (PT), hamza errors (OH) and Taa Marbuta / Haa confusion errors (OT), and approximately 71% of the words in the training data contain no errors. This makes sense since most of the training data comes from the QALB-14 dataset, which is generated mainly by native speakers in the comments sections of news articles.

After we have the error tag distribution, each sentence is then passed through a pipeline where we first use the BERT-based morphological disambiguation in CAMeL tools [19] to return the set of morphological features for each word. Later, we use those features in rule-based and LLM-based error generation. We use *Llama 3.1 8B* [20] and *gpt-4o-mini* from OpenAI in our experiments.

After that, we apply a softmax temperature over the original training error tag distribution to control how much we want to up-sample under-represented errors and down-sample over-represented errors as a hyperparameter. We then apply weighted random sampling to select an error from the list of available errors for each word. Not all errors can be applied to all words; for example, the Definiteness (XF) error cannot be applied to a word if the word is a verb. To closely match the mistakes that a human would make as much as possible, we define rules to exclude errors before sampling an error for each word. We note that this may change the distribution of errors in the final generated dataset from the initial distribution we target. In what follows, we will define how errors are excluded and how they are generated according to each error class.

A. Orthography Errors

We use a rule-based approach for all orthography errors. For hamza errors (OH), to detect whether we can apply the error, we check whether a *hamza* or an *alif-layinna* is in the word. We exclude the first two characters if the word contains the definitive article *alif wa lam* as determined by its CAMeL morphological features. To apply the error, we replace the *alif* letter with the *alif-layinna* letter or vice versa, or replace *alif* with any of its other forms. For *Taa' / Haa'* (OT) confusion errors and *Yaa' / Alif layinna* (OA) confusion errors, we also replace the character with the other if one of the characters is found to induce the respective error. For *Waw Gamaa'a Error* (OW), we remove the last *alif* after the *waw* only if the morphological features returned by the CAMeL BERT disambiguation indicate that the word is masculine plural verb. For *tanween error*, we remove the *tanween* if we find it. For vowel diacritics and their long form letters confusion errors (OS and OG), we replace the diacritic with its long form letter or vice versa. Finally, for character replacement, addition, and omission errors (OR, OD, and OM respectively), we apply the corresponding character transformation to induce the error.

B. Morphology Errors

To induce a morphological inflection error (MI), we use the root of the word extracted by the Camel BERT disambiguator and prompt the LLM to write a different morphological re-inflection for the word. We use a few shot prompt, where we add illustrative examples of how to generate a different inflection of a word given its root. To induce a verb-tense change error (MT), we give the LLM a few shot prompt where we show examples of how to change between different tenses. This process does not succeed in all cases. If the LLM fails to generate the response in the required format for whatever reason, a fail-safe option is implemented to keep the word unchanged.

C. Syntax Errors

For case errors (XC), we also prompt an LLM where we give a few examples by changing words between nominative, accusative, and genitive cases. For definitiveness errors (XF) we rely on the CAMeL BERT disambiguation and a rule-based approach. We first make sure that the word is a noun and then add the definitiveness *Alif w lam* or remove it depending on whether it was there. Finally, for gender and number errors (XG and XN, respectively), we first make sure that the input word is numbered/gendered. Then, we rely on the LLM using a few shot prompt showing examples of how to change between genders in different cases and how to switch between singular, dual, and plural numbers.

D. Semantics Errors

For preposition errors (SW), we first check whether the word is an Arabic preposition or contains a concatenated Arabic preposition. If it is the first case, we replace it with any other Arabic preposition. If it is the second case, we augment the features returned by the CaMEL BERT disambiguation and use the re-inflection feature in CAMeL tools to return a re-inflected word that has a different concatenated preposition. For conjunction errors (SF), we make sure that the word is a conjunction or contains a conjunction letter concatenated in it. If the word is a conjunction, we randomly replace it with another Arabic conjunction or remove it. If the word contains a conjunction letter, we remove it using the CAMeL re-inflection feature.

E. Punctuation, Merge and Split Errors

For punctuation replacement, missing, and unnecessary addition errors (PC, PM, and PT, respectively), we replace, remove, or add punctuation marks to induce the required error. For an incorrect word-split error, we split the word at a random index. Finally, for incorrect word merge errors, we merge the word with the word immediately following it.

IV. Experiments and Analysis

TABLE I
PERFORMANCE COMPARISON OF ARABART FINETUNING WITH AND WITHOUT THE INCLUSION OF SYNTHETIC DATA.

Testset	Finetuning Data	P	R	F1	F0.5
qalb-15-L1	qalb-15	80.67	72.49	76.36	78.89
	+*synth.*	**82.02**	**73.07**	**77.31**	**80.1**
qalb-15-L2	qalb-15	68.04	43.6	53.15	61.18
	+*synth.*	**69.44**	**43.79**	**53.71**	**62.16**
zaebuc	qalb-15	67.15	68.29	67.72	67.38
	+*synth.*	**69.91**	**69.55**	**69.73**	**69.84**
	qalb-15 + zaebuc	89.39	74.19	81.08	85.87
	+*synth.*	**90.45**	**76.92**	**83.14**	**87.38**

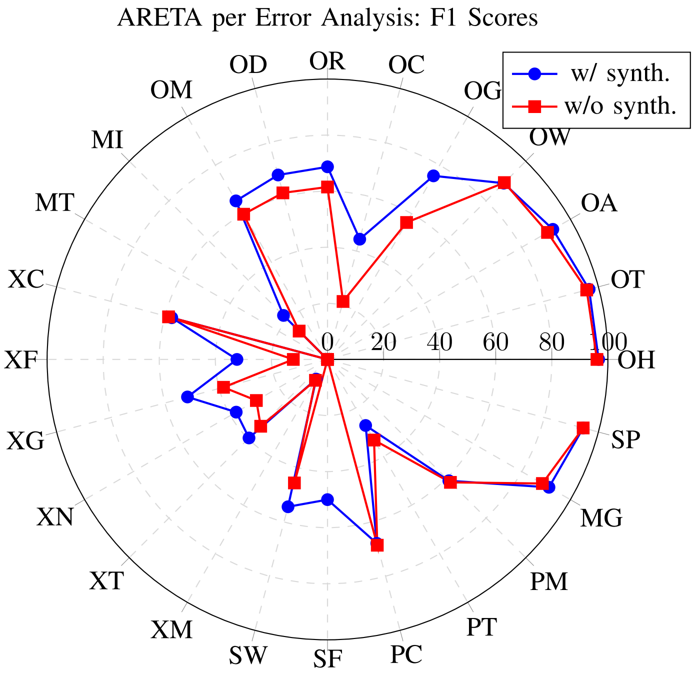

Fig. 1. Per error performance scored using ARETA. We omit error types that have 0 support.

We build our synthetic data using the clean sentence of each pair in the QALB-15 dataset and a sample of 50k sentences

TABLE II
PERFORMANCE COMPARISON BETWEEN ARABART MODEL TRAINED INCLUDING OUR SYNTHETIC DATA AND RELATED WORK.

Testset	System	Punct.				No Punct.			
		P	R	F1	F0.5	P	R	F1	F0.5
qalb-15-L1	Alhafni et al. [11]	**82.6**	71.2	77	**80.3**	90	81.4	85.5	88.2
	Kwon et al. [14]	72.1	**73.6**	72.8	72.4	85.8	72.9	78.8	82.9
	AraBART + *Synth.*	82.1	73.1	**77.3**	80.1	**90.1**	**83.3**	**86.6**	**88.6**
qalb-15-L2	Alhafni et al. [11]	69	**45.4**	**54.7**	**62.5**	70.7	43.6	53.9	62.9
	AraBART + *Synth.*	**69.4**	43.8	53.7	62.2	**72.2**	43.6	**54.3**	**63.8**
ZAEBUC	Alhafni et al. [11]	**85.9**	**73.4**	**79.2**	**83.1**	89.9	**77.8**	**83.4**	87.2
	AraBART + *Synth.*	83.3	72	77.2	80.8	**90.5**	76.9	83.1	**87.4**

of the public Arabic Billion Words corpus [21], which mainly consists of news articles. Since we use those sentences as the target and generate grammatically corrupted input from them, it is important that the datasets selected for synthetic data generation are of high quality.

We used different values of temperatures and different LLMs to build our synthetic data. For QALB-15 grammatically correct sentences, we utilize *LLama-3.1-8B* with a few-shot prompt and test two different temperature settings. The first setup is $t = 20$ applied to all probabilities of the error classes, including unchanged (UC), and this is the more aggressive setup. The second setup is $t = 10$ and keep the percentage of unchanged words the same and just apply the temperature to the percentages of error classes. In our experiments, we found both setups to be beneficial although the more aggressive setup improved the recall more at the expense of a slight drop in precision.

To scale synthetic data generation on the 50k sentences sample of the Arabic Billion Words corpus, we use the aggressive setup with an increased $t = 25$ on 50% of the data and the less aggressive setup on the other 50% of the data. We also change the LLM to *gpt-4o-mini* with a few shot prompt since we have found it to perform better when prompted to induce errors on a word given the error class. For *LLama-3.1-8B*, we needed to add a definition and an example for each error class. However, for *gpt-4o-mini*, a few-shot prompt was sufficient to get the desired result.

For the final synthetic data, we combine the QALB-15 based synthetic data and the Arabic Billion Words corpus based synthetic data. We finetune 2 sets of models. The first set uses only real data, and the other set uses real data and synthetic data. When training with a combination of synthetic and real data, it is a common practice to sample real data more frequently. When we finetune the real and synthetic data together, we sample real data twice as often as the synthetic data.

We run two sets of experiments, one using the QALB-15 shared task dataset and the other using the QALB-15 and ZAEBUC datasets. We use AraBART as a pre-trained model to finetune. For all models, we train for 10 epochs and pick the best checkpoint according to a development set. Table I shows the effect of adding our synthetic dataset to the finetuning dataset. We can see a notable improvement when we include the synthetic dataset compared to when we do not include it during finetuning. To test whether training using our synthetic data improves performance in out-of-domain datasets, we test models finetuned only QALB-15 data as real data on the ZAEBUC testset. Table I shows that including our synthetic data gives a significant improvement when testing on an out-of-domain test set. Using ARETA [9], we compare the performance per error class of AraBART finetuned on QALB-15 data to that of AraBART finetuned on QALB-15 and our synthetic data. As shown in figure1, including our synthetic data helps the model to perform better on the more challenging non-orthographic and non-punctuation errors.

We also compare our AraBART model finetuned using our synthetic data with recent state-of-the-art Arabic GEC models. For comparison, our model tested on QALB-15 testsets includes only QALB-15 training data as real data, and our model tested on the ZAEBUC testset includes QALB-15 and ZAEBUC training data as real data. For the systems from Alhafni et al. [11], we selected the top performing system for each dataset. Notably, these systems incorporate an additional grammatical error detection transformer model to assist with GEC.

Due to observed inconsistencies in reference punctuation in the QALB-15 shared task [3], we also report results without punctuation. Our system achieves performance close to the state-of-the-art and even surpasses it when punctuation marks are excluded from evaluation in the QALB-15 shared task. It also matches the state-of-the-art performance in ZAEBUC when punctuation marks are excluded. In addition, our model does not make use of an external grammatical error detection, which makes it more efficient.

V. CONCLUSION AND FUTURE WORK

We have presented a recipe for generating synthetic data to complement existing Arabic GEC datasets and demonstrated its effectiveness in improving performance on challenging Arabic grammatical error types. While recent progress in Arabic GEC is encouraging, the performance of current GEC systems on complex Arabic grammatical error types leaves more to be desired.

Future work could focus on iteratively refining the distribution of synthetic data based on performance across specific error types on target development sets. Furthermore, integrating rules from similar methods, such as those proposed by Magdy et al. [6] with our hybrid approach, could further enhance the performance and robustness of Arabic GEC systems.

References

[1] K. Darwish, N. Habash, M. Abbas, H. Al-Khalifa, H. T. Al-Natsheh, H. Bouamor, K. Bouzoubaa, V. Cavalli-Sforza, S. R. El-Beltagy, W. El-Hajj, *et al.*, "A panoramic survey of natural language processing in the arab world," *Communications of the ACM*, vol. 64, no. 4, pp. 72–81, 2021.

[2] N. Y. Habash, "Introduction to arabic natural language processing," *Synthesis lectures on human language technologies*, vol. 3, no. 1, pp. 1–187, 2010.

[3] A. Rozovskaya, H. Bouamor, N. Habash, W. Zaghouani, O. Obeid, and B. Mohit, "The second qalb shared task on automatic text correction for arabic," in *Proceedings of the Second workshop on Arabic natural language processing*, pp. 26–35, 2015.

[4] B. Mohit, A. Rozovskaya, N. Habash, W. Zaghouani, and O. Obeid, "The first qalb shared task on automatic text correction for arabic," in *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pp. 39–47, 2014.

[5] N. Habash and D. Palfreyman, "ZAEBUC: An annotated Arabic-English bilingual writer corpus," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, (Marseille, France), pp. 79–88, European Language Resources Association, June 2022.

[6] S. M. Magdy, F. Alwajih, S. Y. Kwon, R. Abdel-Salam, and M. Abdul-Mageed, "Gazelle: An instruction dataset for Arabic writing assistance," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, (Miami, Florida, USA), pp. 16027–16054, Association for Computational Linguistics, Nov. 2024.

[7] L. Long, R. Wang, R. Xiao, J. Zhao, X. Ding, G. Chen, and H. Wang, "On llms-driven synthetic data generation, curation, and evaluation: A survey," *arXiv preprint arXiv:2406.15126*, 2024.

[8] F. Stahlberg and S. Kumar, "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*, (Online), pp. 37–47, Association for Computational Linguistics, Apr. 2021.

[9] R. Belkebir and N. Habash, "Automatic error type annotation for Arabic," in *Proceedings of the 25th Conference on Computational Natural Language Learning*, (Online), pp. 596–606, Association for Computational Linguistics, Nov. 2021.

[10] A. Alfaifi and E. Atwell, "An evaluation of the arabic error tagset v2," in *Proceedings of the AACL 2014-The American Association for Corpus Linguistics conference*, The American Association for Corpus Linguistics, 2014.

[11] B. Alhafni, G. Inoue, C. Khairallah, and N. Habash, "Advancements in Arabic grammatical error detection and correction: An empirical investigation," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, (Singapore), pp. 6430–6448, Association for Computational Linguistics, Dec. 2023.

[12] M. Kamal Eddine, N. Tomeh, N. Habash, J. Le Roux, and M. Vazirgiannis, "AraBART: a pretrained Arabic sequence-to-sequence model for abstractive summarization," in *Proceedings of the The Seventh Arabic Natural Language Processing Workshop (WANLP)*, (Abu Dhabi, United Arab Emirates (Hybrid)), pp. 31–42, Association for Computational Linguistics, Dec. 2022.

[13] E. M. B. Nagoudi, A. Elmadany, and M. Abdul-Mageed, "AraT5: Text-to-text transformers for Arabic language generation," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Dublin, Ireland), pp. 628–647, Association for Computational Linguistics, May 2022.

[14] S. Kwon, G. Bhatia, E. M. B. Nagoudi, and M. Abdul-Mageed, "Beyond English: Evaluating LLMs for Arabic grammatical error correction," in *Proceedings of ArabicNLP 2023*, (Singapore (Hybrid)), pp. 101–119, Association for Computational Linguistics, Dec. 2023.

[15] K. Omelianchuk, V. Atrasevych, A. Chernodub, and O. Skurzhanskyi, "Gector–grammatical error correction: tag, not rewrite," *arXiv preprint arXiv:2005.12592*, 2020.

[16] S. Mahmoud, E. Nabil, and M. Torki, "Automatic scoring of arabic essays: A parameter-efficient approach for grammatical assessment," *IEEE Access*, vol. 12, pp. 142555–142568, 2024.

[17] M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki, "The penn arabic treebank: Building a large-scale annotated arabic corpus," in *NEMLAR conference on Arabic language resources and tools*, vol. 27, pp. 466–467, Cairo, 2004.

[18] A. Alrehili and A. Alhothali, "Tibyan corpus: Balanced and comprehensive error coverage corpus using chatgpt for arabic grammatical error correction," *arXiv preprint arXiv:2411.04588*, 2024.

[19] O. Obeid, N. Zalmout, S. Khalifa, D. Taji, M. Oudah, B. Alhafni, G. Inoue, F. Eryani, A. Erdmann, and N. Habash, "CAMeL tools: An open source python toolkit for Arabic natural language processing," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, (Marseille, France), pp. 7022–7032, European Language Resources Association, May 2020.

[20] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[21] I. A. El-Khair, "1.5 billion words arabic corpus," *arXiv preprint arXiv:1611.04033*, 2016.

APPENDIX

TABLE III
ERROR TYPES STATISTICS IN COMBINED QALB-15 SHARED TASK AND ZAEBUC TRAINING DATASETS.

	Tag	Error Description	Example	Percentage
Orthography (O)	OH	Hamza error	أكثر←اكثر	8.32%
	OT	Confusion in Ha and Ta & Alif-Maqsura	مشاركة←مشاركه	1.47%
	OA	Confusuion in Alif and Ya	على←علي	0.75%
	OW	Confusion in Alif Fariqa	وكانوا←وكانو	0.03%
	ON	Nun and tanwin confusion	ثوبٌ←ثوبن	0%
	OS	Shortening the long vowels	أوقات←أوقَت	0%
	OG	Lengthening the short vowels	نقيمُ←نقيمو	0.18%
	OC	Char. order	تربينا←تبرينا	0.05%
	OR	Char. replacement	وصلنا←مصلنا	1.16%
	OD	Additional character(s)	يدوم←يعدوم	0.95%
	OM	Missing character(s)	سائلين←سالين	0.58%
Morphology (M)	MI	Word inflection	عارف←معروف	0.56%
	MT	Verb tense	أفرحتني←تفرحني	0.02%
Syntax (X)	XC	Case	رائع←رائعاً	0.66%
	XF	Definiteness	السن←سن	0.16%
	XG	Gender	الغربية←الغربي	0.14%
	XN	Number	أفكاري←فكرتي	18.5%
	XT	Unnecessary word	على←Null	0.21%
	XM	Missing word	Null←على	0.75%
Semantics (S)	SW	Word selection error	من←عن	0.5%
	SF	Conjunction error	فسبحان←سبحان	0.14%
Punctuation (P)	PC	Punctuation confusion	المتوسط،←المتوسط.	1.05%
	PT	Unnecessary punctuation	العام←العام،	8.61%
	PM	Missing punctuation	العظيم،←العظيم	0.53%
Merge	MG	Words are merged	ذهبت البارحة←ذهبتالبارحة	0.87%
Split	SP	Words are split	المحادثات←المحا دثات	0.72%
No Error	UC	Unchanged / No Error	المدرسة←المدرسة	71.42%