Return_Zero at Bhashabhrom: Bangla Grammatical Error Detection Leveraging Transformer-based Token Classification

Ridwanul Hasan Tanvir*†, Shayekh Bin Islam* and Sihat Afnan*
*Department of Computer Science and Engineering, BUET
Dhaka, Bangladesh
†Corresponding author: ridwanulhasan.tanvir@gmail.com
shayekh.bin.islam@gmail.com, sihatafnan15.9.1997@gmail.com

Abstract—Bhashabhrom: Bangla Grammatical Error Detection Challenge is a task of detecting sub-strings of a Bangla text that contain grammatical, punctuation, or spelling errors, which is crucial for developing an automated Bangla typing assistant. This paper represents the Phase 1 winning method for this task. Our approach involves breaking down the task as a Token Classification problem and utilizing state-of-the-art transformer-based models. Finally, we combine the output of these models to generate a more reliable and comprehensive result. Our system achieves Levenshtein distance of 1.0540 and 1.0336 in public (50%) and private (50%) leaderboards respectively, ranking first among 50 teams in Phase 1.

Index Terms—Bangla, natural language processing, grammatical error, spelling error, deep learning

I. Introduction

Typing assistance has become increasingly important in today's digital age, especially with the rise of grammar checking systems. Users expect typing assistance tools to not only detect and correct grammatical errors but also provide suggestions to improve their writing skills. With the help of predictive text, spell checkers, and auto-correct features, typing assistance tools can help users to type accurately and efficiently, reducing the time needed to correct errors. These features can also help users to expand their vocabulary and improve their grammar, leading to better writing skills and effective communication. With the rise of AI and machine learning, typing assistance tools such as Grammarly are becoming more advanced, helping users to improve their writing skills and communicate effectively.

Errors in Bangla text can be due to spelling, punctuation or grammar. The spelling error itself can occur in various forms [1] shown in Table I. Examples of errors similar to ERRANT [2] error classes are shown in Table II. Alam et. al. [3] approaches the punctuation correction problem only considering commas, periods, and question marks for ASR applications.

TABLE I
Types of Spelling Errors

Error Type	Example
Cognitive Error	পরবাস $\rightarrow$ পরবাশ
Visual Error	দেবতা $\rightarrow$ দেরতা
Typo Insertion	চুল্লি $\rightarrow$ চুলল্লি
Typo Deletion	দুর্বার $\rightarrow$ দুর্বর
Typo Transposition	ঘাটতি $\rightarrow$ ঘাতটি
Typo (Avro) Substitution	চেয়ার $\rightarrow$ চেয়াএ
Typo (Bijoy) Substitution	ঘুর্ণি $\rightarrow$ ঘুর্ষি
Run-on Error	ত্রিভুবন $\rightarrow$ ত্রিভুবনঅষ্টক
Split-word Error (Random)	মিহি $\rightarrow$ মি হি
Split-word Error (Left)	ঘোলাটে $\rightarrow$ ঘোলা টে
Split-word Error (Right)	অশান্তি $\rightarrow$ অ শান্তি
Split-word Error (Both)	শ্রেণিকক্ষ $\rightarrow$ শ্রেণি কক্ষ
Homonym Error	বর্ষা $\rightarrow$ বর্শা

TABLE II
Errors by ERRANT Classification

Error Type	Example
Spelling	পরনির্ভরশীল $\rightarrow$ ফরনির্ভরশীল
Orthography	ব্যবসা প্রতিষ্ঠান $\rightarrow$ ব্যবসাপ্রতিষ্ঠান
Punctuation	। $\rightarrow$!
Noun Inflection	অধিবাসীরা $\rightarrow$ অধিবাসী
Pronoun	আমি $\rightarrow$ আমরা
Verb Tense	যাবে $\rightarrow$ যায়
Adjective Form	মৃত (স্ত্রী) $\rightarrow$ মৃতা (স্ত্রী)
Subject-Verb Agreement	(সে) খায় $\rightarrow$ (সে) খাই
Conjunction	কিন্তু $\rightarrow$ এবং
Literary Register	পড়ে $\rightarrow$ পড়িয়া

II. Related Works

Numerous NLP techniques have been devised to address sentence-level errors, with statistical and rule-based methods being the most prevalent. Rule-based methods involve creating language-specific rules to tackle errors, while statistical approaches are favored for their language-independence, making them more widely used than rule-based techniques.

In Bangla NLP, B. B. Chaudhuri has implemented an approximate string matching algorithm for detecting non-word errors [4], while N. UzZaman and M. Khan used a direct dictionary lookup method to handle misspelled word errors [5], and Abdullah and Rahman employed it to detect typographical and cognitive phonetic errors [6]. P. Mandal and B. M. M. Hossain proposed a method based on the PAM clustering algorithm, which also did not address semantic errors [7]. A few works have been done at the semantic level. N. Hossain introduced a model that utilizes n-grams to check whether a word is correctly used in a sentence [8], while K. M. Hasan, M. Hozaifa, and S. Dutta developed a rule-based method for detecting grammatical semantic errors in simple sentences [9].

In Bhashabhrom, the task is a span detection problem in general. One of the notable span detection problem is SemEval-2021 task 5: Toxic spans detection [10], where the goal is to detect toxic spans within English passages.

The solutions for the Toxic spans detection task formalize the task as a token-level sequence labeling (SL) problem [11] [12] [13] [14] [15] [16] [17], as a span detection problem [11] [17] and as a dependency parsing problem [15]. Generating pseudo labels from external datasets [12] [16] for semi-supervised learning is a notable part of some solutions. Here, the winning models are various transformer-based models along with LSTM and CRF. To combine the predictions of various models, they employ the union or the intersection of the predicted spans.

III. Methodology

We formalize the task as a four-class token classification problem similar to the well-known BIO tagging scheme. Then, we employ transformer-based models along with LSTM and CRF. Next, we ensemble and post-process the model predictions to generate the final output.

A. Data Pre-processing

1) Normalization: Following the normalization scheme of BanglaBERT, we normalize punctuations and characters with multiple Unicode representations [18] to avoid [UNK] during tokenization.

2) Class Labelling: To prepare the dataset for a four-class classification problem (no error(O), begin error(B), inside error(I), missing after(M)), we extract the character indices from the normalized sentence with $ enclosed annotations corresponding to each class, for example:

$$\overbrace{\text{ডিম ভাজির সাথে}}^{\text{O}}\ \overbrace{\text{\$পুরা\$}}^{\text{B}}\ \overbrace{\text{\$মাছ}}^{\text{B}}\ \overbrace{\text{টাই\$}}^{\text{I}}\ \overbrace{\text{খেলাম\$\$}}^{\text{M}}$$

We incorporate a label for the [CLS] token when the proportion of toxic offsets in the text exceeds 30%. This enables the system to be trained on a proxy text classification objective [17].

B. Token Classification Models

1) Transformer-based Token Classification Models: The Token Classification Model is composed of an ELECTRA-based model, specifically BanglaBERT-base and BanglaBERT-large [19], and a classification layer that is applied to each final token embedding to predict the error class of each token as shown in Figure 1.

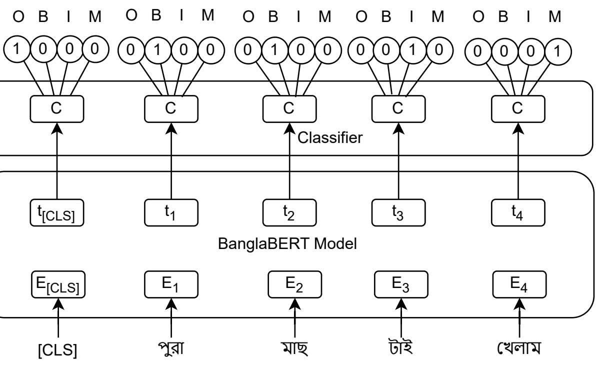

Fig. 1. Transformer-based Token Classification Models

2) LSTM-CRF Token Classification Models: A popular approach for Named-Entity Recognition tasks is to combine Conditional Random Fields (CRF) with transformer-based models, as demonstrated in recent studies [20] [21] [17]. In our approach, we utilize BanglaBERT-based models that incorporate a single BiLSTM layer and a CRF layer. During training, the CRF loss is applied, while Viterbi Decoding is performed during prediction.

C. Post-processing

1) Error Span Generation: Biasing the method to prefer the original tokens, unless the model is very confident about an error, produces superior results [22]. We choose the confidence threshold that performs best in the dev set, often ranging from 0.7 to 0.8.

Once our model generates the token-level labels, we further process the output to transform it into an array of character offsets that correspond to tokens with error classes. To achieve this, we first map each token to its offset span during tokenization and then extract the character offsets associated with all the error tokens.

2) Reverse Normalization: The models generate error spans with respect to the normalized text, whereas the task expects the final output to be with respect to the text without any modification (such as normalization). To solve this problem, we calculate the minimum Levenshtein edit-distance-based alignment mapping between the normalized and the original texts using edit_distance_align function from the NLTK library and apply some rule-based corrections to get the best alignment. Thus, we get the final error spans in the expected form.

D. Ensemble Learning

Utilizing the predictions from the top few checkpoints and averaging the results helps obtain superior classification scores [23] [17]. Building on this approach, we also aggregate the predicted spans from different checkpoints within a model, as well as across different models [12], using union or intersection methods.

E. Deterministic Error Detection

1) Punctuation Error: If there is any space before periods, commas, question marks or exclamation marks, we manually mark that span with spaces as an error. Similarly, if the sentence does not end with punctuation, we report a punctuation missing error.

2) Spelling Error: We collect a database of about one million spelling errors from DPCSpell [1]. We filter out spelling errors that are absent from the online Bangla dictionary. Next, to make the models refrain from indentifying named entities as spelling errors, we further filter out the spelling errors that are absent from the word collections of the titles of the Bangla Wikipedia pages. Now, we get a filtered collection of one million spelling errors. Finally, we mark a word as error if the word exits in the filtered spelling error collection and the word is not a named entity. We use the NER model from the open-source BNLP library as a named entity classifier.

IV. Experiments

A. Dataset

The competition dataset for Phase 1 comprises around 25000 texts extracted from various online sources. The training dataset is released in two phases with a total of around 20000 texts and the test dataset contains 5000 texts. We split the training dataset int to train and dev sets for evaluation purposes using an 80:20 split stratified by the number of error spans.

B. Evaluation Metrics

For this task, we employ the Levenshtein Distance of the predicted string and ground truth to evaluate our models (lower is better). The Levenshtein distance between two strings a,b (of length $|a|$ and $|b|$ respectively) is given by $\operatorname{lev}(a, b)$ where

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) & \text{if } a[0] = b[0], \\ 1 + \min \begin{cases} \operatorname{lev}\big(\operatorname{tail}(a), b\big) \\ \operatorname{lev}\big(a, \operatorname{tail}(b)\big) \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) \end{cases} & \text{otherwise,} \end{cases}$$

C. Implementation Details

For all pre-trained transformer-based models and tokenizers, we use HuggingFace's transformers [24] in PyTorch framework. We finetune BanglaBERT-base and BanglaBERT-large [19] for 30 epochs with batch size of 8. We use an AdamW optimizer [25] with learning rate of 2e-5 and weight decay of 0.01. We use a linear learning rate decay with warmup ratio of 0.1. Evaluating the models multiple times during an epoch increases the chance of getting better validation performance [26]. Hence, we evaluate the model after 500 gradient steps. During tokenization, we use the maximum length of 384.

Label smoothing improves the accuracy of Inception networks on ImageNet [27] and NLP tasks [28] [12] by serving as a form of regularization. This involves assigning a small probability to non-ground-truth labels, which can prevent the models from being too confident about their predictions and improve generalization. Label smoothing has proven to be a useful alternative to the standard cross entropy loss and has been widely adopted to address over-confidence [29] [30] [31], improve the model calibration [32], and de-noise incorrect labels [33]. We use label smoothing factor of 0.1 for BanglaBERT-base and 0.2 for BanglaBERT-large.

V. Results

A. Pre-trained Models

We initially experiment on different transformers models that are multilingual and for Bangla specifically in order to select the right backbones on a binary classification task of detecting whether a token is error or not, shown in Table III. Here, all the models are trained using standard cross entropy loss with punctuation post-processing and no normalization. We find that BanglaBERT models are best candidates for this task.

TABLE III
Comparison of Transformers Models on Private Test Set

Model	Levenstein Distance
XLM-RoBERTa-base	1.394
DeBERTa-V3-large	1.3552
BanglaBERT-base	1.2120
BanglaBERT-large	1.1844

B. LSTM-CRF

We do not observe any performance improvement by using LSTM-CRF on top of transformer-based pretrained models in the dev set. Hence, we employ BanglaBERT-base and BanglaBERT-large as the final soulution.

C. Deterministic Error Detection

We check for extra spaces before punctuation (space fix) and missing punctuation at the end (end fix). The models fail to capture theses errors an we notice performance improvement by applying these modification on the model output that are presented in Table IV. We also notice slight improvement by manual spelling error detection using our one million spelling error database.

TABLE IV
Effectiveness of deterministic punctuation error detection on the private test set

Model	Levenstein Distance
BanglaBERT-base	1.2948
BanglaBERT-base+space fix	1.246
BanglaBERT-base+space fix+end fix	1.212

D. Ensemble Strategy

We explore the effectiveness span union or intersection methods to ensemble different models as presented in Table V and Table VI. The results suggest that intersection approaches outperform corresponding union and single checkpoint approaches, whereas union approaches perform worse than single checkpoints. These findings imply that the individual checkpoints may be predicting additional offsets that are identified as errors. Hence, We take the intersection of three best checkpoints of BanglaBERT-base and BanglaBERT-large and combine the two predictions by intersection again to achieve the best Levenstein score in test set.

TABLE V
Effectiveness of ensemble I on private test set

Type	Levenstein Distance
BanglaBERT-base+large Union	1.2524
BanglaBERT-base+large Intersection	1.144
BanglaBERT-large only	1.2212

TABLE VI
Effectiveness of Ensemble II on Public Test Set

Model	Levenstein Distance
Single-checkpoint	1.0648
Three-checkpoints	1.054

E. Label Smoothing

Label Smoothing stops the model from overfitting and modelling noise. We find label smoothing to be beneficial to get better performance as shown in Table VII.

TABLE VII
Private test set results for label smoothing

Type	Levenstein Distance
BanglaBERT-large+standard CE	1.164
BanglaBERT-large+smoothing factor 0.2	1.1588

F. Thresholding

Figure 2 demonstrates the effect of confidence thresholding for two checkpoints of the BanglaBERT model in the dev set. We observe that the model performs best near to a threshold of 0.8, hence we choose this threshold for BanglaBERT models during inference which boosts the test set performance as shown in Table VIII.

TABLE VIII
Results of thresholding on the private test set

Type	Levenstein Distance
BanglaBERT-large+threshold 0.0	1.1892
BanglaBERT-large+threshold 0.8	1.1588

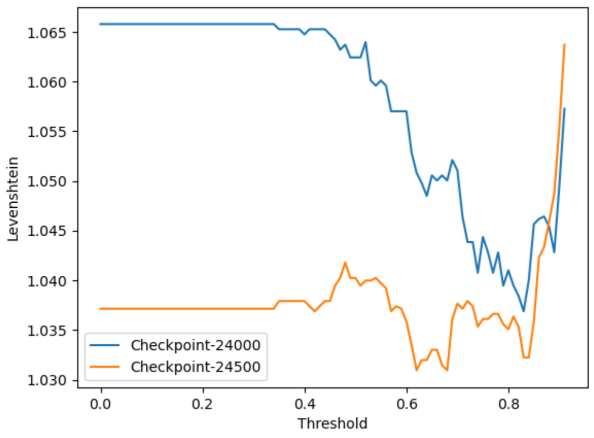

Fig. 2. Effect of confidence thresholding on the dev Set

G. Unicode Normalization

Table IX shows the effectiveness of union normalization. We observe good performance boost when we use the normalization and the reverse normalization properly. For all the models, we use this scheme.

TABLE IX
Results of Unicode normalization on the private test set

Type	Levenstein Distance
BanglaBERT-large without normalization	1.130
BanglaBERT-large with normalization	1.084

H. The Final System

The final system is an intersection-ensemble of three best checkpoints of BanglaBERT-base and BanglaBERT-large. We apply all the aforementioned techniques with the hyper-parameters that perform best in the dev set and the test set.

VI. Discussion

We generate synthetic datasets from the Prothom-Alo scrapes available online to simulate real grammar errors [34]. Though training with this additional data slightly improves the dev set metric, it hurts the test set performance. The cause for this contrast could be attributed to the inconsistent distribution of data in the dev and test sets.

The dataset size is small with respect to the model complexity of the tranformer models. For this reason, overfitting occurs easily if we do not apply label smoothing, thresholding and learning rate scheduling to regularize and help the model reach a generalized solution.

Previous works [13] [15] report performance boost by employing LSTM-CRF with transformer-based models in span detection tasks, but in contrast we find vanilla transformer-based models to perform better. This can happen as we focus on tuning the hyper-parameters for different proposed modification with respect to vanilla transformer-based models only. Extensive hyper-parameter search and optimization strategies can potentially make LSTM-CRF augmented models demonstrate their full potential.

To model the missing errors (empty spans after a character), an alternative approach can be using character-level embedding instead of token level embedding and formalizing this type of error as a separate classification problem apart from non-empty spans detection. We have used a separate classification head for this error but do not observe much improvement.

VII. Conclusion

In this work, we present our approach to Bhashabhrom: Bangla Grammatical Error Detection Challenge. We achieve the best scores in the public and private test set leaderboards using a transformer-based ensembles with a bag of tricks. In future, we intend to enhance our model as follows:

- Employ self-training with in-domain unlabeled data.
- Combine self-training with feature-based learning to learn a more robust model.
- Use Fast Gradient Method (FGM) as adversarial training strategy.

References

[1] M. H. Bijoy, N. Hossain, S. Islam, and S. Shatabda, "Dpcspell: A transformer-based detector-purificator-corrector framework for spelling error correction of bangla and resource scarce indic languages," arXiv preprint arXiv:2211.03730, 2022.

[2] C. Bryant, M. Felice, and E. Briscoe, "Automatic annotation and evaluation of error types for grammatical error correction." Association for Computational Linguistics, 2017.

[3] T. Alam, A. Khan, and F. Alam, "Punctuation restoration using transformer models for high-and low-resource languages," in Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020), 2020, pp. 132–142.

[4] B. B. Chaudhuri, "Reversed word dictionary and phonetically similar word grouping based spell-checker to bangla text," in Proc. LESAL Workshop, Mumbai, 2001.

[5] N. UzZaman and M. Khan, "A comprehensive bangla spelling checker," 2006.

[6] A. Abdullah and A. Rahman, "A generic spell checker engine for south asian languages," in Conference on Software Engineering and Applications (SEA 2003), 2003, pp. 3–5.

[7] P. Mandal and B. M. Hossain, "Clustering-based bangla spell checker," in 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR). IEEE, 2017, pp. 1–6.

[8] N. H. Khan, G. C. Saha, B. Sarker, and M. H. Rahman, "Checking the correctness of bangla words using n-gram," International Journal of Computer Application, vol. 89, no. 11, 2014.

[9] K. A. Hasan, M. Hozaifa, and S. Dutta, "Detection of semantic errors from simple bangla sentences," in 2014 17th International Conference on Computer and Information Technology (ICCIT). IEEE, 2014, pp. 296–299.

[10] J. Pavlopoulos, J. Sorensen, L. Laugier, and I. Androutsopoulos, "Semeval-2021 task 5: Toxic spans detection," in Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021), 2021, pp. 59–69.

[11] Q. Zhu, Z. Lin, Y. Zhang, J. Sun, X. Li, Q. Lin, Y. Dang, and R. Xu, "Hitsz-hlt at semeval-2021 task 5: Ensemble sequence labeling and span boundary detection for toxic span detection," in Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021), 2021, pp. 521–526.

[12] V. A. Nguyen, T. M. Nguyen, H. Q. Dao, and Q. H. Pham, "S-nlp at semeval-2021 task 5: An analysis of dual networks for sequence tagging," in Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021), 2021, pp. 888–897.

[13] C. Wang, T. Liu, and T. Zhao, "Hitmi&t at semeval-2021 task 5: integrating transformer and crf for toxic spans detection," in Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), 2021, pp. 870–874.

[14] R. Chen, J. Wang, and X. Zhang, "Ynu-hpcc at semeval-2021 task 5: Using a transformer-based model with auxiliary information for toxic span detection," in Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), 2021, pp. 841–845.

[15] S. Ghosh and S. Kumar, "Cisco at semeval-2021 task 5: What's toxic?: Leveraging transformers for multiple toxic span extraction from online comments," arXiv preprint arXiv:2105.13959, 2021.

[16] A. Bansal, A. Kaushik, and A. Modi, "Iitk@ detox at semeval-2021 task 5: Semi-supervised learning and dice loss for toxic spans detection," arXiv preprint arXiv:2104.01566, 2021.

[17] G. Chhablani, A. Sharma, H. Pandey, Y. Bhartia, and S. Suthaharan, "Nlrg at semeval-2021 task 5: toxic spans detection leveraging bert-based token classification and span prediction techniques," arXiv preprint arXiv:2102.12254, 2021.

[18] T. Hasan, A. Bhattacharjee, K. Samin, M. Hasan, M. Basak, M. S. Rahman, and R. Shahriyar, "Not low-resource anymore: Aligner ensembling, batch filtering, and new datasets for Bengali-English machine translation," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Online: Association for Computational Linguistics, Nov. 2020, pp. 2612–2623. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-main.207

[19] A. Bhattacharjee, T. Hasan, W. Ahmad, K. S. Mubasshir, M. S. Islam, A. Iqbal, M. S. Rahman, and R. Shahriyar, "BanglaBERT: Language model pretraining and benchmarks for low-resource language understanding evaluation in Bangla," in Findings of the Association for Computational Linguistics: NAACL 2022. Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 1318–1327. [Online]. Available: https://aclanthology.org/2022.findings-naacl.98

[20] F. Souza, R. Nogueira, and R. Lotufo, "Portuguese named entity recognition using bert-crf," arXiv preprint arXiv:1909.10649, 2019.

[21] D. Jurkiewicz, Ł. Borchmann, I. Kosmala, and F. Graliński, "Applicaai at semeval-2020 task 11: On roberta-crf, span cls and whether self-training helps them," arXiv preprint arXiv:2005.07934, 2020.

[22] D. Alikaniotis and V. Raheja, "The unreasonable effectiveness of transformer language models in grammatical error correction," arXiv preprint arXiv:1906.01733, 2019.

[23] H. Chen, S. Lundberg, and S.-I. Lee, "Checkpoint ensembles: Ensemble methods from a single training process," arXiv preprint arXiv:1710.03282, 2017.

[24] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz et al., "Transformers: State-of-the-art natural language processing," in Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations, 2020, pp. 38–45.

[25] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," arXiv preprint arXiv:1711.05101, 2017.

[26] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," arXiv preprint arXiv:2002.06305, 2020.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.

[28] E. Zhu and J. Li, "Boundary smoothing for named entity recognition," arXiv preprint arXiv:2204.12031, 2022.

[29] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8697–8710.

[30] J. Chorowski and N. Jaitly, "Towards better decoding and language model integration in sequence to sequence models," arXiv preprint arXiv:1612.02695, 2016.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.

[32] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" Advances in neural information processing systems, vol. 32, 2019.

[33] M. Lukasik, S. Bhojanapalli, A. Menon, and S. Kumar, "Does label smoothing mitigate label noise?" in International Conference on Machine Learning. PMLR, 2020, pp. 6448–6458.

[34] C. R. Rahman, M. Rahman, S. Zakir, M. Rafsan, and M. E. Ali, "Bspell: A cnn-blended bert based bengali spell checker," arXiv preprint arXiv:2208.09709, 2022.