Bangla Grammatical Error Detection Leveraging Transformer-based Token Classification

Ridwanul Hasan Tanvir
Computer Science & Engineering
Bangladesh University of Engineering and Technology (BUET)
ridwanul@ra.cse.buet.ac.bd

Shayekh Bin Islam
Computer Science & Engineering
Bangladesh University of Engineering and Technology (BUET)
1705009@ugrad.cse.buet.ac.bd

Abstract

Bangla is the seventh most spoken language by a total number of speakers in the world, and yet the development of an automated grammar checker in this language is an understudied problem. Bangla grammatical error detection 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. 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 and apply rule-based post-processing to generate a more reliable and comprehensive result. Our system is evaluated on a dataset consisting of over 25,000 texts from various sources. Our best model achieves a Levenshtein distance score of 1.04. Finally, we provide a detailed analysis of different components of our system.

1 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 (Bijoy et al., 2022). (Alam et al., 2020) approaches the punctuation correction problem only considering commas, periods, and question marks for ASR applications.

2 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 (Chaudhuri, 2001), while N. UzZaman and M. Khan used a direct dictionary lookup method to handle misspelled word errors (UzZaman and Khan, 2006), and Abdullah and Rahman employed it to detect typographical and cognitive phonetic errors (Abdullah and Rahman, 2003). P. Mandal and B. M. M. Hossain proposed a method based on the PAM clustering algorithm, which also did not address semantic errors (Mandal and Hossain, 2017). 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 (Khan et al., 2014), while K. M. Hasan, M. Hozaifa, and S. Dutta developed a rule-based method for detecting grammatical semantic errors in simple sentences (Hasan et al., 2014).

The task of Bangla grammatical error detection is a span detection problem in general. One of the notable span detection problem is SemEval-2021 task 5: Toxic spans detection (Pavlopoulos et al., 2021), 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 la-

beling (SL) problem (Zhu et al., 2021; Nguyen et al., 2021; Wang et al., 2021; Chen et al., 2021; Ghosh and Kumar, 2021; Bansal et al., 2021; Chhablani et al., 2021), as a span detection problem (Zhu et al., 2021; Chhablani et al., 2021) and as a dependency parsing problem (Ghosh and Kumar, 2021). Generating pseudo labels from external datasets (Nguyen et al., 2021; Bansal et al., 2021) 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.

3 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.

3.1 Data Pre-processing

3.1.1 Normalization

Following the normalization scheme of BanglaBERT, we normalize punctuations and characters with multiple Unicode representations (Hasan et al., 2020) to avoid [UNK] during tokenization.

3.1.2 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:

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 (Chhablani et al., 2021).

3.2 Token Classification Models

3.2.1 Transformer-based Token Classification Models

The Token Classification Model is composed of an ELECTRA-based model, specifically BanglaBERT-base and BanglaBERT-large (Bhattacharjee et al., 2022), 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.

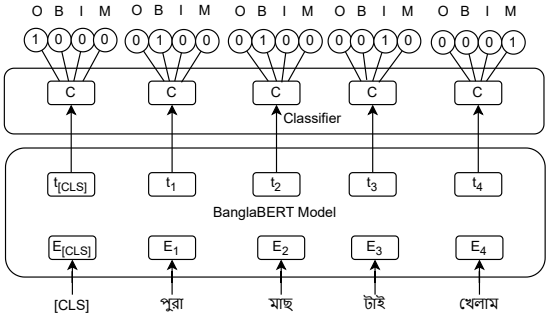

Figure 1: Transformer-based Token Classification Models

3.2.2 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 (Souza et al., 2019; Jurkiewicz et al., 2020; Chhablani et al., 2021). In our approach, we utilize BanglaBERT-based models that incorporate a single BiLSTM layer and a CRF layer as shown in Figure 2.. During training, the CRF loss is applied, while Viterbi Decoding is performed during prediction.

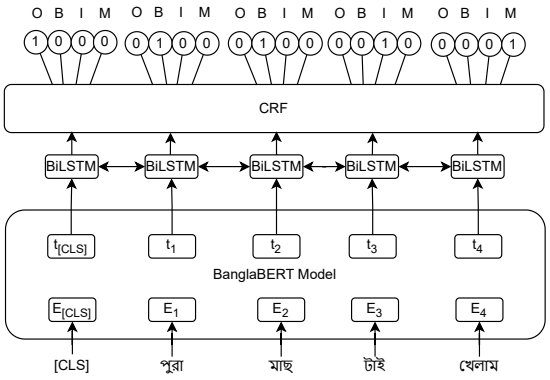

Figure 2: LSTM-CRF Token Classification Models

3.3 Post-processing

3.3.1 Error Span Generation

Biasing the method to prefer the original tokens, unless the model is very confident about an error, produces superior results (Alikaniotis and Raheja,

2019). 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.

3.3.2 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.

3.4 Ensemble Learning

Utilizing the predictions from the top few checkpoints and averaging the results helps obtain superior classification scores (Chen et al., 2017; Chhablani et al., 2021). Building on this approach, we also aggregate the predicted spans from different checkpoints within a model, as well as across different models (Nguyen et al., 2021), using union or intersection methods.

3.5 Deterministic Error Detection

3.5.1 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.

3.5.2 Spelling Error

We collect a database of about one million spelling errors from DPCSpell (Bijoy et al., 2022). We filter out spelling errors that are absent from the online Bangla dictionary. Next, to make the models refrain from identifying 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 an 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.

4 Experiments

4.1 Dataset

We collect the publicly available dataset of Bangla Grammatical Error Detection Challenge (Murad et al., 2023) that comprises around 25000 texts extracted from various online sources. The training dataset contains a total of around 20000 texts and the test dataset contains 5000 texts. We split the training dataset into train and dev sets for evaluation purposes using an 80:20 split stratified by the number of error spans.

4.2 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

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

4.3 Implementation Details

For all pre-trained transformer-based models and tokenizers, we use HuggingFaces transformers (Wolf et al., 2020) in the PyTorch framework. We finetune BanglaBERT-base and BanglaBERT-large (Bhattacharjee et al., 2022) for 30 epochs with batch size of 8. We use an AdamW optimizer (Loshchilov and Hutter, 2017) 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 (Dodge et al., 2020). 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 (Szegedy et al., 2016) and NLP tasks (Zhu and Li, 2022; Nguyen

et al., 2021) 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 (Zoph et al., 2018; Chorowski and Jaitly, 2016; Vaswani et al., 2017), improve the model calibration (Müller et al., 2019), and de-noise incorrect labels (Lukasik et al., 2020). We use a label smoothing factor of 0.1 for BanglaBERT-base and 0.2 for BanglaBERT-large.

5 Results

5.1 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 an error or not, shown in Table 1. Here, all the models are trained using standard cross-entropy loss with punctuation post-processing and no normalization. We find that BanglaBERT models are the best candidates for this task.

Table 1: Comparison of Transformers Models on Test Set

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

5.2 LSTM-CRF

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

5.3 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 these errors an we notice performance improvement by applying these modifications on the model output that are presented in Table 2. We also notice a slight improvement in manual spelling error detection using our one million spelling error database.

Table 2: Effectiveness of deterministic punctuation error detection on the test set

Model	LD
BanglaBERT-base	1.2948
BanglaBERT-base+space fix	1.246
BanglaBERT-base+space fix+end fix	**1.212**

5.4 Ensemble Strategy

We explore the effectiveness of span union or intersection methods to ensemble different models as presented in Table 3 and Table 4. 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 the three best checkpoints of BanglaBERT-base and BanglaBERT-large and combine the two predictions by intersection again to achieve the best Levenstein score in the test set.

Table 3: Effectiveness of ensemble I on test set

Type	LD
BanglaBERT-base+large Union	1.2524
BanglaBERT-base+large Intersection	**1.144**
BanglaBERT-large only	1.2212

Table 4: Effectiveness of Ensemble II on Test Set

Model	LD
Single-checkpoint	1.0648
Three-checkpoints	**1.054**

5.5 Label Smoothing

Label Smoothing stops the model from overfitting and modeling noise. We find label smoothing to be beneficial to get better performance as shown in Table 5.

5.6 Thresholding

Figure 3 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

Table 5: Test set results for label smoothing

Type	LD
BanglaBERT-large+standard CE	1.164
BanglaBERT-large+smoothing 0.2	1.1588

threshold of 0.8, hence we choose this threshold for BanglaBERT models during inference which boosts the test set performance as shown in Table 6.

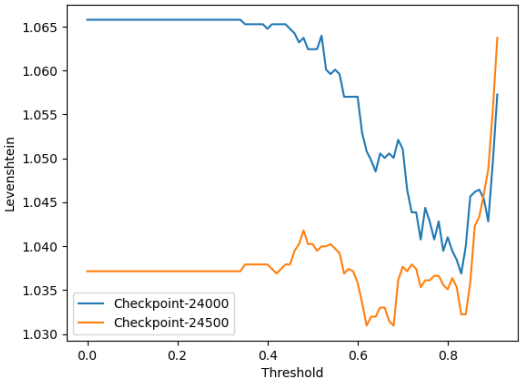

Figure 3: Effect of confidence thresholding on the dev Set

Table 6: Results of thresholding on the test set

Type	LD
BanglaBERT-large+threshold 0.0	1.1892
BanglaBERT-large+threshold 0.8	**1.1588**

5.7 Unicode Normalization

Table 7 shows the effectiveness of union normalization. We observe a performance boost when we use the normalization and the reverse normalization properly. For all the models, we use this scheme.

5.8 The Final System

The final system is an intersection ensemble of the 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.

6 Discussion

We generate synthetic datasets from the Prothom-Alo scrapes available online to simulate real grammar errors (Rahman et al., 2022). Though training

Table 7: Results of Unicode normalization on the test set

Type	LD
BanglaBERT-large without normalization	1.130
BanglaBERT-large with normalization	**1.084**

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 transformer 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 (Wang et al., 2021; Ghosh and Kumar, 2021) 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 modifications 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 as shown in Figure 4. We have used a separate classification head for this error but do not observe much improvement.

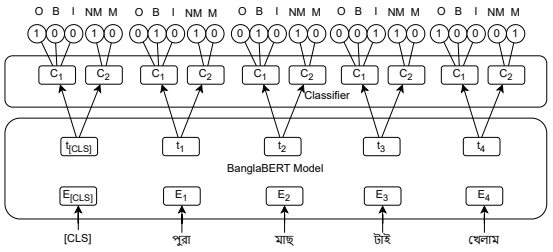

Figure 4: Separate head for missing errors

7 Conclusion

In this work, we present our approach to solving the Bangla grammatical error detection problem that outperforms multiple baselines. We formalize the problem as a sequence labeling problem and apply different transformer-based models and feature-based models. BanglaBERT, an ELECTRA model pre-trained on clean Bangla corpus, is at the heart of our system. We also find that clever choice of ensemble, loss function, and rule-based post-processing significantly improves the machine learning-based systems. Our model is effective in detecting various kinds of grammar errors in Bangla and will motivate deep learning-based approaches to solve this complex problem.

In future, we intend to enhance our model by employing self-training with in-domain unlabeled data, combining self-training with feature-based learning to learn a more robust model, and using Fast Gradient Method (FGM) as an adversarial training strategy. Moreover, we will evaluate our system for the Bangla grammar error correction problem.

References

ABA Abdullah and Ashfaq Rahman. 2003. A generic spell checker engine for south asian languages. In *Conference on Software Engineering and Applications (SEA 2003)*, pages 3–5.

Tanvirul Alam, Akib Khan, and Firoj Alam. 2020. 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)*, pages 132–142.

Dimitrios Alikaniotis and Vipul Raheja. 2019. The unreasonable effectiveness of transformer language models in grammatical error correction. *arXiv preprint arXiv:1906.01733*.

Archit Bansal, Abhay Kaushik, and Ashutosh Modi. 2021. Iitk@ detox at semeval-2021 task 5: Semi-supervised learning and dice loss for toxic spans detection. *arXiv preprint arXiv:2104.01566*.

Abhik Bhattacharjee, Tahmid Hasan, Wasi Ahmad, Kazi Samin Mubasshir, Md Saiful Islam, Anindya Iqbal, M. Sohel Rahman, and Rifat Shahriyar. 2022. BanglaBERT: Language model pretraining and benchmarks for low-resource language understanding evaluation in Bangla. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1318–1327, Seattle, United States. Association for Computational Linguistics.

Mehedi Hasan Bijoy, Nahid Hossain, Salekul Islam, and Swakkhar Shatabda. 2022. Dpcspell: A transformer-based detector-purificator-corrector framework for spelling error correction of bangla and resource scarce indic languages. *arXiv preprint arXiv:2211.03730*.

Bidyut Baran Chaudhuri. 2001. Reversed word dictionary and phonetically similar word grouping based spell-checker to bangla text. In *Proc. LESAL Workshop, Mumbai*.

Hugh Chen, Scott Lundberg, and Su-In Lee. 2017. Checkpoint ensembles: Ensemble methods from a single training process. *arXiv preprint arXiv:1710.03282*.

Ruijun Chen, Jin Wang, and Xuejie Zhang. 2021. Ynuhpcc 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)*, pages 841–845.

Gunjan Chhablani, Abheesht Sharma, Harshit Pandey, Yash Bhartia, and Shan Suthaharan. 2021. Nlrg at semeval-2021 task 5: toxic spans detection leveraging bert-based token classification and span prediction techniques. *arXiv preprint arXiv:2102.12254*.

Jan Chorowski and Navdeep Jaitly. 2016. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Sreyan Ghosh and Sonal Kumar. 2021. Cisco at semeval-2021 task 5: What's toxic?: Leveraging transformers for multiple toxic span extraction from online comments. *arXiv preprint arXiv:2105.13959*.

KM Azharul Hasan, Muhammad Hozaifa, and Sanjoy Dutta. 2014. Detection of semantic errors from simple bangla sentences. In *2014 17th International Conference on Computer and Information Technology (ICCIT)*, pages 296–299. IEEE.

Tahmid Hasan, Abhik Bhattacharjee, Kazi Samin, Masum Hasan, Madhusudan Basak, M. Sohel Rahman, and Rifat Shahriyar. 2020. 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)*, pages 2612–2623, Online. Association for Computational Linguistics.

Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. 2020. Applicaai at semeval-2020 task 11: On roberta-crf, span cls and

whether self-training helps them. *arXiv preprint arXiv:2005.07934*.

Nur Hossain Khan, Gonesh Chandra Saha, Bappa Sarker, and Md Habibur Rahman. 2014. Checking the correctness of bangla words using n-gram. *International Journal of Computer Application*, 89(11).

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Michal Lukasik, Srinadh Bhojanapalli, Aditya Menon, and Sanjiv Kumar. 2020. Does label smoothing mitigate label noise? In *International Conference on Machine Learning*, pages 6448–6458. PMLR.

Prianka Mandal and BM Mainul Hossain. 2017. Clustering-based bangla spell checker. In *2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 1–6. IEEE.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? *Advances in neural information processing systems*, 32.

Md Boktiar Mahbub Murad, Sushmit, and Tasnim Nishat Islam. 2023. Apurba presents bhashabhrom: Eee day 2023 datathon.

Viet Anh Nguyen, Tam Minh Nguyen, Huy Quang Dao, and Quang Huu Pham. 2021. 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)*, pages 888–897.

John Pavlopoulos, Jeffrey Sorensen, Léo Laugier, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection. In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*, pages 59–69.

Chowdhury Rafeed Rahman, MD Rahman, Samiha Zakir, Mohammad Rafsan, and Mohammed Eunus Ali. 2022. Bspell: A cnn-blended bert based bengali spell checker. *arXiv preprint arXiv:2208.09709*.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2019. Portuguese named entity recognition using bert-crf. *arXiv preprint arXiv:1909.10649*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Naushad UzZaman and Mumit Khan. 2006. A comprehensive bangla spelling checker.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Chenyi Wang, Tianshu Liu, and Tiejun Zhao. 2021. 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)*, pages 870–874.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 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.

Enwei Zhu and Jinpeng Li. 2022. Boundary smoothing for named entity recognition. *arXiv preprint arXiv:2204.12031*.

Qinglin Zhu, Zijie Lin, Yice Zhang, Jingyi Sun, Xiang Li, Qihui Lin, Yixue Dang, and Ruifeng Xu. 2021. 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)*, pages 521–526.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710.