On the Robustness of Language Encoders against Grammatical Errors

Fan Yin[1], **Quanyu Long**[2], **Tao Meng**[3], **and Kai-Wei Chang**[3]
[1]Peking University
[2]Shanghai Jiao Tong University
[3]University of California, Los Angeles
1600012975@pku.edu.cn;
oscar.long@sjtu.edu.cn;
tmeng@cs.ucla.edu;
kwchang@cs.ucla.edu

Abstract

We conduct a thorough study to diagnose the behaviors of pre-trained language encoders (ELMo, BERT, and RoBERTa) when confronted with natural grammatical errors. Specifically, we collect real grammatical errors from non-native speakers and conduct adversarial attacks to simulate these errors on clean text data. We use this approach to facilitate debugging models on downstream applications. Results confirm that the performance of all tested models is affected but the degree of impact varies. To interpret model behaviors, we further design a linguistic acceptability task to reveal their abilities in identifying ungrammatical sentences and the position of errors. We find that fixed contextual encoders with a simple classifier trained on the prediction of sentence correctness are able to locate error positions. We also design a cloze test for BERT and discover that BERT captures the interaction between errors and specific tokens in context. Our results shed light on understanding the robustness and behaviors of language encoders against grammatical errors.

1 Introduction

Pre-trained language encoders have achieved great success in facilitating various downstream natural language processing (NLP) tasks (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2019b). However, they usually assume training and test corpora are clean and it is unclear how the models behave when confronted with noisy input. Grammatical error is an important type of noise since it naturally and frequently occurs in natural language, especially in spoken and written materials from non-native speakers. Dealing with such a noise reflects model robustness in representing language and grammatical knowledge. It would also have a positive social impact if language encoders can model texts from non-native speakers appropriately.

Recent work on evaluating model's behaviors against grammatical errors employs various methods, including (1) manually constructing minimal edited pairs on specific linguistic phenomena (Marvin and Linzen, 2018; Goldberg, 2019; Warstadt et al., 2019a,b); (2) labeling or creating acceptability judgment resources (Linzen et al., 2016; Warstadt and Bowman, 2019; Warstadt et al., 2019a); and (3) simulating noises for a specific NLP task such as neural machine translation (Lui et al., 2018; Anastasopoulos, 2019), sentiment classification (Baldwin et al., 2017). These studies either focus on specific phenomena and mainly conduct experiments on designated corpora or rely heavily on human annotations and expert knowledge in linguistics. In contrast, our work automatically simulates natural occurring data and various types of grammatical errors and systematically analyzes how these noises affect downstream applications. This holds more practical significance to understand the robustness of several language encoders against grammatical errors.

Specifically, we first propose an effective approach to simulating diverse grammatical errors, which applies black-box adversarial attack algorithms based on real errors observed on NUS Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013), a grammatical error correction benchmark. This approach transforms clean corpora into corrupted ones and facilitates debugging language encoders on downstream tasks. We demonstrate its flexibility by evaluating models on four language understanding tasks and a sequence tagging task.

We next quantify model's capacities of identifying grammatical errors by probing individual layers of pre-trained encoders through a linguistic acceptability task. We construct separate datasets for eight error types. Then, we freeze encoder layers

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 3386–3403

and add a simple classifier on top of each layer to predict the correctness of input texts and locate error positions. This probing task assumes if a simple classifier behaves well on a designated type of error, then the encoder layer is likely to contain knowledge of that error (Conneau et al., 2017; Adi et al., 2017).

Finally, we investigate how models capture the interaction between grammatical errors and contexts. We use BERT as an example and design an unsupervised cloze test to evaluate its intrinsic functionality as a masked language model (MLM).

Our contributions are summarized as follows:

1. We propose a novel approach to simulating various grammatical errors. The proposed method is flexible and can be used to verify the robustness of language encoders against grammatical errors.
2. We conduct a systematic analysis of the robustness of language encoders and enhance previous work by studying the performance of models on downstream tasks with various grammatical error types.
3. We demonstrate: (1) the robustness of existing language encoders against grammatical errors varies; (2) the contextual layers of language encoders acquire stronger abilities in identifying and locating grammatical errors than token embedding layers; and (3) BERT captures the interaction between errors and specific tokens in context, in particular the neighboring tokens of errors.

The code to reproduce our experiments are available at: https://github.com/uclanlp/ProbeGrammarRobustness

2 Related Work

Probing Pre-trained Language Encoders The recent success of pre-trained language encoders across a diverse set of downstream tasks has stimulated significant interest in understanding their advantages. A portion of past work on analyzing pre-trained encoders is mainly based on clean data. As mentioned in Tenney et al. (2019a), these studies can be roughly divided into two categories: (1) designing controlled tasks to probe whether a specific linguistic phenomenon is captured by models (Conneau et al., 2018; Peters et al., 2019; Tenney et al., 2019b; Liu et al., 2019a; Kim et al., 2019), or (2) decomposing the model structure and exploring what linguistic property is encoded (Tenney et al., 2019a; Jawahar et al., 2019; Clark et al., 2019). However, these studies do not analyze how grammatical errors affect model behaviors.

Our work is related to studies on analyzing models with manually created noise. For example, Linzen et al. (2016) evaluate whether LSTMs capture the hierarchical structure of language by using verbal inflection to violate subject-verb agreement. Marvin and Linzen (2018) present a new dataset consisting of minimal edited pairs with the opposite linguistic acceptability on three specific linguistic phenomena and use it to evaluate RNN's syntactic ability. Goldberg (2019) adjusts previous method to evaluate BERT. Warstadt et al. (2019a) further compare five analysis methods under a single phenomenon. Despite the diversity in methodology, these studies share common limitations. First, they employ only a single or specific aspects of linguistic knowledge; second, their experiments are mainly based on constructed datasets instead of real-world downstream applications. In contrast, we propose a method to cover a broader range of grammatical errors and evaluate on downstream tasks. A concurrent work (Warstadt et al., 2019b) facilitates diagnosing language models by creating linguistic minimal pairs datasets for 67 isolate grammatical paradigms in English using linguist-crafted templates. In contrast, we do not rely heavily on artificial vocabulary and templates.

Synthesized Errors To evaluate and promote the robustness of neural models against noise, some studies manually create new datasets with specific linguistic phenomena (Linzen et al., 2016; Marvin and Linzen, 2018; Goldberg, 2019; Warstadt et al., 2019a). Others have introduced various methods to generate synthetic errors on clean downstream datasets, in particular, machine translation corpora. Belinkov and Bisk (2018); Anastasopoulos (2019) demonstrate that synthetic grammatical errors induced by character manipulation and word substitution can degrade the performance of NMT systems. Baldwin et al. (2017) augment original sentiment classification datasets with syntactically (reordering) and semantically (word substitution) noisy sentences and achieve higher performance. Our method is partly inspired by Lui et al. (2018), who synthesize semi-natural ungrammatical sentences by maintaining confusion matrices for five simple error types.

Another line of studies uses black-box adversarial attack methods to create adversarial examples

for debugging NLP models (Ribeiro et al., 2018; Jin et al., 2019; Alzantot et al., 2018; Burstein et al., 2019). These methods create a more challenging scenario for target models compared to the above data generation procedure. Our proposed simulation benefits from both adversarial attack algorithms and semi-natural grammatical errors.

3 Method

We first explain how we simulate ungrammatical scenarios. Then, we describe target models and the evaluation design.

3.1 Grammatical Error Simulation

Most downstream datasets contain only clean and grammatical sentences. Despite that recent language encoders achieve promising performance, it is unclear if they perform equally well on text data with grammatical errors.

Therefore, we synthesize grammatical errors on clean corpora to test the robustness of language encoders. We use a controllable rule-based method to collect and mimic errors observed on NUCLE following previous work (Lui et al., 2018; Sperber et al., 2017) and apply two ways to introduce errors to clean corpora: (1) we sample errors based on the frequency distribution of NUCLE and introduce them to plausible positions; (2) inspired by the literature of adversarial attacks (Ribeiro et al., 2018; Jin et al., 2019; Alzantot et al., 2018), we conduct search algorithms to introduce grammatical errors that causing the largest performance drop on a given downstream task.

Mimic Error Distribution on NUCLE We first describe how to extract the error distribution on NUCLE (Dahlmeier et al., 2013). NUCLE is constructed with naturally occurring data (student essays at NUS) annotated with error tags. Each ungrammatical sentence is paired with its correction that differs only in local edits. The two sentences make up a *minimal edited pair*. An example is like:

1. Will the child blame the parents after he **growing** up? ×
2. Will the child blame the parents after he **grows** up? ✓

NUCLE corpus contains around 59,800 sentences with average length 20.38. About 6% of tokens in each sentence contain grammatical errors. There are 27 error tags, including `Prep` (indicating preposition errors), `ArtOrDet` (indicating article or determiner errors), `Vform` (indicating incorrect verb form) and so forth.

We consider eight frequently-occurred, token-level error types in NUCLE as shown in Table 1.

These error types perturb a sentence in terms of syntax (`SVA`, `Worder`), semantics (`Nn`, `Wchoice`, `Trans`) and both (`ArtOrDet`, `Prep`, `Vform`), and thus cover a wide range of noise in natural language. Then, we construct a confusion set for each error type based on the observation on NUCLE. Each member of a confusion set is a token. We assign a weight w_{ij} between token t_i and t_j in the same set to indicate the probability that t_i will be replaced by t_j. In particular, for `ArtOrDet`, `Prep` and `Trans`, the confusion set consists of a set of tokens that frequently occur as errors or corrections on NUCLE. For each token t_i in the set, we compute w_{ij} based on how many times t_i is replaced by t_j in minimal edited pairs on NUCLE.

Notice that we add a special token $\emptyset$ to represent deletion and insertion. For `Nn`, when we find a noun, we add it and its singular (SG) or plural (PL) counterpart to the set. For `SVA`, when we find a verb with present tense, we add it and its third-person-singular (3SG) or non-third (not 3SG) counterpart to the set. For `Worder`, we exchange the position of an adverb with its neighboring adjective, participle or modal. For `Vform`, we use NLTK (Bird and Loper, 2004) to extract present, past, progressive, and perfect tense of a verb and add to the set. For `Wchoice`, we select ten synonyms of a target word from WordNet. The substitution weight is set to be uniform for both `Vform` and `Wchoice`.

Grammatical Error Introduction We introduce errors in two ways. The first is called *probabilistic transformation*. Similar to Lui et al. (2018), we first obtain the parse tree of the target sentence using the Berkeley syntactic parser (Petrov et al., 2006). Then, we sample an error type from the error type distribution estimated from NUCLE and randomly choose a position that can apply this type of error according to the parse tree. Finally, we sample an error token based on the weights from the confusion set of the sampled error type and introduce the error token to the selected position.

However, *probabilistic transformation* only represents the average case. To debug and analyze the robustness of language encoders, we consider another more challenging setting – *worst-case transformation*, where we leverage search algorithms

Error type	Error Description	Confusion Set
ArtOrDet	Article/determiner errors	{ a, an, the, ∅}
Prep	Preposition errors	{ on, in, at, from, for, under, over, with, into, during, until, against, among, throughout, to, by, about, like, before, across, behind, but, out, up, after, since, down, off, of, ∅}
Trans	Link words/phrase errors	{and, but, so, however, as, that, thus, also, because, therefore, if, although, which, where, moreover, besides, of, ∅}
Nn	Noun number errors	{SG, PL}
SVA	Subject-verb agreement errors	{3SG, not 3SG}
Vform	Verb form errors	{Present, Past, Progressive, Perfect}
Wchoice	Word choice errors	{Ten synonyms from WordNet Synsets}
Worder	Word positions errors	{Adverb w/ Adjective, Participle, Modal}

Table 1: The target error types and the corresponding confusion sets.

from the black-box adversarial attack to determine error positions. More concretely, we obtain an operation set for each token in a sentence by considering all possible substitutions based on all confusion sets. Note that some confusion sets are not applicable, for example the confusion set of Nn to a verb. Each operation in the operation set is to replace the target token or to change its position. Then, we apply a searching algorithm to select operations from these operation sets that change the prediction of the tested model and apply them to generate error sentences. Three search algorithms are considered: *greedy search*, *beam search*, and *genetic algorithm*.

Greedy search attack is a two-step procedure. First, we evaluate the importance of tokens in a sentence. The importance of a token is represented by the likelihood decrease on the model prediction when it is deleted. The larger the decrease is, the more important the token is. After comparing all tokens, we obtain a sorted list of tokens in descending order of their importance. Then, we walk through the list. For each token in the list, we try out all operations from the operation set associated with that token and then practice the operation that degrades the likelihood of the model prediction the most. We keep repeating step two until the prediction changes or a budget (e.g., number of operations per sentence) is reached.

Beam search is similar to *greedy search*. The only difference is that when we walk through the sorted list of tokens, we maintain a beam with fixed size k that contains the top k operation streams with the highest global degradation.

Genetic algorithm is a population-based iterative method for finding more suitable examples. We start by randomly selecting operations to build a generation and then use a combination of crossover and mutation to find better candidates. We refer the readers to Alzantot et al. (2018) for details of the genetic algorithm in adversarial attack. Comprehensive descriptions of all methods are found in Appendix C.

3.2 Target Models

We evaluate the following three pre-trained language encoders. Detailed descriptions of models and training settings are in Appendix B.

ELMo (Peters et al., 2018) is a three-layer LSTM-based model pre-trained on the bidirectional language modeling task on 1B Word Benchmark (Chelba et al., 2014). We fix ELMo as a contextual embedding and add two layers of BiLSTM with attention mechanism on top of it.

BERT (Devlin et al., 2019) is a transformer-based (Vaswani et al., 2017) model pre-trained on masked language modeling and next sentence prediction tasks. It uses 16GB English text and adapts to downstream tasks by fine-tuning. We use *BERT-base-cased* for Named Entity Recognition (NER) and *BERT-base-uncased* for other tasks and perform task-specific fine-tuning.

RoBERTa (Liu et al., 2019b) is a robustly pre-trained BERT model using larger pre-training data (160GB in total), longer pre-training time, the dynamic masking strategy and other optimized pre-

training methods. We use *RoBERTa-base* and perform task-specific fine-tuning.

3.3 Evaluation Methods

We design the following three evaluation methods to systematically analyze how language encoders are affected by grammatical errors in input.

Simulate Errors on Downstream Tasks Using the simulation methods discussed in Section §3.1, we are able to perform evaluation on existing benchmark corpora. In our experiments, we consider the target models independently. The whole procedure is: given a dataset, the target model is first trained (fine-tuned) and evaluated on the clean training and development set. Then, we discard those wrongly predicted examples from the development set and apply simulation methods to perturb each remaining example. We compute the attack success rate (attacked examples / all examples) as an indicator of model robustness against grammatical errors. The smaller the rate is, the more robust a model is.

Linguistic Acceptability Probing We design a linguistic acceptability probing task to evaluate each individual type of error. We consider two aspects: (1) if the model can tell whether a sentence is grammatically correct or not (i.e., a binary classification task); (2) if the model can locate error positions in the token-level. We fix the target model and train a self-attention classifier to perform both probing tasks.

Cloze test for BERT We design an unsupervised cloze test to evaluate the masked language model component of BERT based on minimal edited pairs. For each minimal pair that differs only in one token, we quantify how the probability of predicting a single masked token in the rest of the sentence affected by this grammatical error. This method analyzes how error token affects clean context, which is complementary to Goldberg (2019) who focuses on `SVA` error and discusses how clean contexts influence the prediction of the masked error token.

4 How Grammatical Errors Affect Downstream Performance?

In this section, we simulate grammatical errors and analyze performance drops on downstream tasks.

We compare ELMo, BERT, RoBERTa and a baseline model InferSent (Conneau et al., 2017).

	Infersent	ELMo	BERT	RoBERTa
MRPC	75.42	80.30	86.48	89.88
MNLI-m	68.62	74.91	83.77	87.70
MNLI-mm	69.12	75.50	84.80	87.40
QNLI	77.39	78.23	90.58	92.50
SST-2	83.14	90.37	92.08	94.72
NER	-	91.21	95.20	95.45

Table 2: Original performance of the target models on language understanding and sequential tagging tasks.

Datasets We use four language understanding datasets: MRPC (Dolan and Brockett, 2005), MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), and SST-2 (Socher et al., 2013) from GLUE (Wang et al., 2019a) and a sequence tagging benchmark: CoNLL-2013 for NER. Detailed descriptions of these corpora are in Appendix A. We do not use other datasets from GLUE since they are either small in size or only contain short sentences.

Attack Settings For all tasks, we limit the maximum percentage of allowed modifications in a sentence to be 15% of tokens, which is a reasonable rate according to the statistics estimated from the real data. As shown in Table 3, the *worst-case transformation* only modifies around 9% of tokens overall under such a limitation. For MNLI and QNLI, we only modify the second sentence, i.e., hypothesis and answer, respectively. For MRPC, we only modify the first sentence. We do not apply the genetic algorithm to MNLI and QNLI due to their relatively large number of examples in the development sets, which induce an extremely long time for attacking. For NER, we keep the named entities and only modify the remaining tokens.

Results and Discussion Table 2 presents the test performance of four target models on the standard development set of each task. Table 3 summarizes the attack success rates on language understanding tasks, the decreases of F1 score on NER, and the mean percentage of modified tokens (number in brackets). All numbers are formatted in percentage.

As shown in Table 3, with the *probabilistic transformation*, the attack success rates fall between 2% (RoBERTa, QNLI) and 10% (ELMo, MRPC). With the *worst-case transformation*, we obtain the highest attacked rate of 81.1% (ELMo, genetic algorithm, MRPC) and an average attacked rate across all tasks of 29% by perturbing only around 9% of tokens. This result confirms that all models are influenced by ungrammatical inputs. NER task is

Model	Alg.	MRPC	MNLI (m/mm)	QNLI	SST-2	NER
Infersent	dist.	6.51 (14.53)	8.30 (13.98) / 8.80 (14.23)	4.76 (12.53)	5.79 (14.38)	-
	greedy	53.42 (9.02)	36.52 (10.35) / 40.71 (10.06)	44.92 (7.61)	43.44 (8.02)	-
	beam	54.39 (9.08)	36.66 (10.37) / 40.87 (10.06)	45.16 (7.62)	43.86 (8.03)	-
	genetic	79.15 (8.60)	-	-	59.86 (8.39)	-
BiLSTM	dist.	9.99 (14.53)	7.76 (13.98) / 7.83 (14.23)	5.34 (12.53)	4.64 (14.38)	3.29 (13.75)
+ ELMo	greedy	60.84 (8.19)	29.58 (10.28) / 32.92 (9.89)	39.12 (7.25)	37.55 (8.24)	17.81 (7.67)
+ Attn	beam	61.49 (8.29)	29.74 (10.29) / 33.12 (9.91)	40.38 (7.33)	38.32 (8.32)	18.33 (7.85)
	genetic	81.14 (7.41)	-	-	59.25 (8.25)	39.78 (8.19)
BERT	dist.	3.69(14.53)	6.59 (13.98) / 6.95 (14.23)	2.33 (12.53)	4.73 (14.38)	3.07 (13.75)
	greedy	31.25 (7.95)	28.76 (10.28) / 32.04 (10.01)	25.43 (7.38)	33.54 (7.96)	17.12 (7.51)
	beam	31.81 (8.01)	29.03 (10.30) / 32.44 (10.04)	26.42 (7.48)	34.28 (8.01)	18.27 (7.74)
	genetic	59.01 (8.84)	-	-	58.53 (7.83)	38.83(7.64)
RoBERTa	dist.	3.04 (14.53)	5.66 (13.98) / 5.88(14.23)	1.92 (12.53)	3.53 (14.38)	2.52 (13.75)
	greedy	20.45 (8.11)	20.65 (10.43) / 21.47 (10.02)	19.82 (7.18)	31.06 (8.20)	15.84 (8.12)
	beam	20.73(8.14)	20.89 (10.44) / 21.91 (10.06)	20.52 (7.29)	31.91 (8.27)	16.51 (7.47)
	genetic	38.93 (9.17)	-	-	56.41 (8.39)	35.11(7.55)

Table 3: Results of evaluating the robustness of models on downstream tasks. Each column represents a dataset and each row represents a victim model with the attack algorithm (dist. means *probabilistic transformation*). In each cell, we show the mean attack success rate (in percentage) and the mean percentage of modified words (number in the bracket) over the dataset.

	BERT			RoBERTa		
	MRPC	MNLI	SST	MRPC	MNLI	SST
Prep	16	178	36	15	103	43
Art/Det	5	270	20	7	228	28
Wchoice	93	1129	233	64	772	195
Vform	8	231	26	9	314	37
SVA	57	538	83	31	388	83
Nn	14	128	13	3	84	13
Worder	0	62	28	0	43	28
Trans	5	70	25	5	31	25

Table 4: Numbers of times each error type is chosen in successful attacks. We find that `Wchoice` and `SVA` are more harmful.

in general harder to be influenced by grammatical errors. In terms of the *probabilistic transformation*, the drop of F1 scores ranges from 2% to 4%. For the *worst-case transformation*, the highest drop for NER is 18.33% (ElMo, beam search).

Considering different target models, we observe that the impact of grammatical errors varies among models. Specifically, RoBERTa exhibits a strong robustness against the impact of grammatical errors, with consistently lower attack success rates (20.28% on average) and F1 score decreases (17.50% on average) across all tasks, especially on MRPC and MNLI. On the other hand, BERT, ELMo, and InferSent experience an average attack rate of 26.03%, 33.06%, 36.07% respectively on NLU tasks. Given the differences in pre-training strategies, we speculate that pre-training with more data might benefit model robustness against noised data. This speculation is consistent with (Warstadt et al., 2019b), where the authors also give a lightweight demonstration on LSTM and Transformer-XL (Dai et al., 2019) with varying training data. We leave a further exploration of this speculation and a detailed analysis of model architecture to future work.

Note that in the experiment setting, for each model, we follow the literature to compute the attack success rate only on the instances where the model makes correct predictions. Therefore, the attack success rates across different models are not comparable. To compare the robustness of different encoders, we further examine the attack success rates on the common part in the development set that all the models make correct predictions. We find that the overall trend is similar to that in Table 3. For example, the greedy attack success rates of RoBERTa, BERT, and ELMo on MRPC and SST-2 are 14.4%, 22.1%, 46.8%, and 28.2%, 30.0%, 33.9% respectively.

To better understand the effect of grammatical errors, we also analyze (1) which error type harms the performance most, (2) how different error rates affect the performance. For the first question, we represent the harm of an error type by the total time it is chosen in successful greedy attack examples. We conduct experiments to analyze BERT and RoBERTa on the development sets of MRPC, MNLI-m, and SST-2 as shown in Table

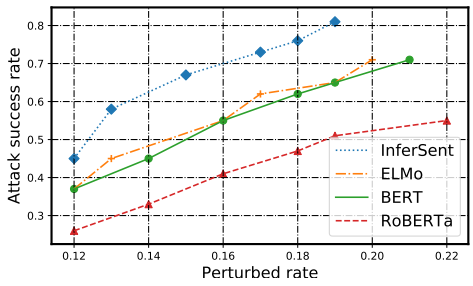

Figure 1: Attack success rate when the numbers of modified tokens in a sentence increase.

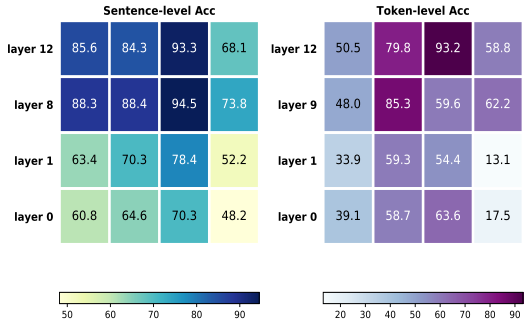

Figure 2: Probing four layers of BERT on four error types. The left side shows the accuracy of the binary linguistic acceptability task. The right side shows the accuracy of locating error positions. Each row represents a specific layer, and each column represents a type of errors, `ArtOrDet`, `Nn`, `SVA`, `Worder` from left to right. Full results are given in Appendix D

4. Among all, `Wchoice` is the most harmful type while `Worder` the least. `SVA` ranks the second most harmful type. Notice that though `Nn` changes a token in a similar way with `SVA` (both adding or dropping -s or -es in most cases), they have different influences to the model. As for errors related to function words, `Prep` plays a more important role in general but `ArtOrDet` harms MNLI more.

For the second one, we increase the allowed modifications of greedy attack from 15% to 45% of tokens in one sentence, resulting the actual percentage of modified tokens under 20%. We evaluate all models on the development set of MNLI-m. Results are shown in Fig 1. We find that all attack success rates grow almost linearly as we increase modifications. ELMo and BERT perform almost the same while InferSent grows faster at the beginning and RoBERTa grows slower when reaching the end. The average attack success rate comes to 70% when the error rate is around 20%.

5 To What Extent Models Identify Grammatical Errors?

Our goal in this section is to assess the ability of the pre-trained encoders in identifying grammatical errors. We use a binary linguistic acceptability task to test the model ability in judging the grammatical correctness of a sentence. We further study whether the model can precisely locate error positions, which reflects the token-level ability.

Data We construct separate datasets for each specific type of grammatical error. For each dataset, we extract 10,000 sentences whose lengths fall within 10 to 60 tokens from 1B Word Benchmark (Chelba et al., 2014). Then, we introduce the target error type to half of these sentences using *probabilistic transformation* and keep the error rate over each dataset around 3% (resulting in one or two errors in each sentence). Sentences are split into training (80%), development (10%) and test (10%).

Models We study individual layers of ELMo (2 layers), BERT-base-uncased (12 layers) and RoBERTa-base (12 layers). In particular, we fix each layer and attach a trainable self-attention layer on top of it to obtain a sentence representation. The sentence representation is fed into a linear classifier to output the probability of whether the sentence is linguistically acceptable. See details about the self-attention layer and the linear classifier in Appendix B.3. We next extract the top two positions with the heaviest weights from the trained self-attention layer. If the positions with error token are included, we consider the errors are correctly located by the model in the token-level. This suggests whether contextual encoders are providing enough information for the classifier to identify error locations. For comparisons, we also evaluate the input embedding layer (non-contextualized, layer 0) of each model as a baseline. We compute accuracy for both sentence-level and token-level evaluations.

Results and Discussion We visualize the results of four layers of BERT on four error types, `ArtOrDet`, `Nn`, `SVA`, and `Worder` in Fig 2. Complete results of all layers and other error types are in Appendix D. We find that the mean sentence-level accuracy of the best contextual layers of BERT, ELMo, and RoBERTa across error types are 87.8%, 84.3%, and 90.4%, respectively, while input embedding layers achieve 64.7%, 65.8%, and 66.0%. In token-level, despite trained only on the

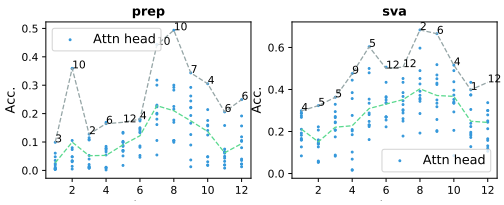

Figure 3: The accuracy of each attention head of BERT on token-level evaluation. The grey line stands for the best performing heads. The green line stands for the average performance of heads in one layer.

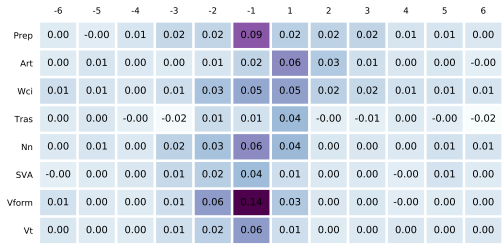

Figure 4: Probing BERT as an MLM. Each row represents a target error type. Each column represents the distance from the error position. Each number represents the mean likelihood drop over all pairs. We find that specific tokens are affected more by error tokens.

prediction of whether a sentence is acceptable, the mean accuracy of classifiers upon the best layers of BERT, ELMo, and RoBERTa are 79.3%, 63.3%, and 80.3%, compared to 48.6%, 18.7%, and 53.4% of input embedding layers. The two facts indicate that these pre-trained encoder layers possess stronger grammatical error detecting and locating abilities compared to input embedding layers.

We also observe patterns related to a specific model. Specifically, middle layers (layers 7-9) of BERT are better at identifying errors than lower or higher layers, as shown in Fig 2. But higher layers of BERT locate errors related to long-range dependencies and verbs such as `SVA` and `Vform` more accurately. To further investigate BERT's knowledge of error locations. We conduct the same token-level evaluation to the 144 attention heads in BERT. Results for `Prep` and `SVA` are visualized in Fig 3. We find that even in a completely unsupervised manner, some attention heads results for 50%-60% accuracy in locating errors. Consistent with self-attention layers, attention heads from middle layers perform the best. See Appendix F for all error types.

Due to space limit, we present results of RoBERTa and ELMo in Appendix D and summarize the observations in the following. RoBERTa exhibits better ability in detecting and locating errors in lower layers compared to BERT and achieves the best performance in top layers (layers 10-11). It is also good at capturing verb and dependency errors. On the other hand, the first layer of ELMo consistently gives the highest sentence-level classification accuracy. But its best performing layer in locating errors depends on the error type and varies between the first and the second layer. In particular, The second layer of ELMo exhibits strong ability in locating `Nn` and outperforms BERT in accuracy. This is surprising given the fact that `Nn` is not obvious with character embeddings from layer 0 of ELMo. We further notice that for all models, `Worder` is the hardest type to detect in the sentence-level and `ArtOrDet` and `Worder` are the hardest types to locate in the token-level. We hypothesize this is related to the locality of these errors which induces a weak signal for models to identify them. Appendix E demonstrates some examples of the token-level evaluation of BERT.

6 How BERT Captures the Interaction between Tokens When Errors Present

We aim to reveal the interaction between grammatical errors and their nearby tokens through studying the masked language model (MLM) component of BERT. We investigate BERT as it is a typical transformer-based encoder. Our analysis can be extended to other models.

Experimental Settings We conduct experiments on minimal edited pairs from NUCLE. We extract pairs with error tags `ArtOrDet`, `Prep`, `Vt`, `Vform`, `SVA`, `Nn`, `Wchoice`, `Trans` and keep those that only have one token changed. This gives us eight collections of minimal edited pairs with sizes of 586, 1525, 1817, 943, 2513, 1359, 3340, and 452, respectively.

Given a minimal edited pair, we consider tokens within six-token away from the error token. We replace the same token in the grammatical and ungrammatical sentence with `[MASK]` one at a time and use BERT as an MLM to predict its likelihood. Then we compute the likelihood drop in the ungrammatical sentence and obtain the average drop over all minimal edited pairs.

Results and Discussion Results are visualized in Fig 4. In general, We find that the decrease of likelihood on specific positions are greater than others

✓	This would thus reduce the financial burden of **this group** of people based on their income ceilings .				
×	This would thus reduce the financial burden of **these group** of people based on their income ceilings .				
	burden	of	this (these)	group	of
	0.01	0.09	-	0.41	0.02
✓	The inexpensive fuel cost and the sheer volume of energy produced by **a** nuclear **reactor** far outweighs the cost of research and development .				
×	The inexpensive fuel cost and the sheer volume of energy produced by **the** nuclear **reactor** far outweighs the cost of research and development .				
	produced	by	a (the)	nuclear	reactor
	0.05	-0.02	-	0.31	0.42

Table 5: Examples with `ArtOrDet`. We show the minimal edit pairs and the likelihood decrease of each token within two tokens away from the errors. Wrong determiners and their corrections are marked in red. The heads in determiner-noun dependencies are marked in blue. As shown in the table, the heads in determiner-noun dependencies get the largest likelihood decrease.

in the presence of errors. Given the fact that certain dependencies between tokens such as subject-verb and determiner-noun dependencies are accurately modeled by BERT as demonstrated in prior work (Jawahar et al., 2019), we suspect that the presence of an error token will mostly affect its neighboring tokens (both in terms of syntactic and physical neighbors). This is consistent with our observation in Fig 4 that in the case of `SVA` where a subject is mostly the preceding token of a verb (although agreement attractors can exist between subject and verb), the proceeding tokens of error positions get the largest likelihood decreases overall. In the case of `ArtOrDet` where an article or a determiner can be an indicator and a dependent of the subsequent noun, predicting the next tokens of error positions becomes much harder. We provide two running examples with `ArtOrDet` in Table 5 to further illustrate this point.

7 Adversarial Training

Finally, we explore a data augmentation method based on the proposed grammatical error simulations. We apply the greedy search algorithm to introduce grammatical errors to the training examples of a target task and retrain the model on the combination of original examples and the generated examples. We take the MRPC (Dolan and Brockett, 2005) dataset as an example to demonstrate the results. We augment the training set of MRPC with different proportions of adversarial examples, fine-tune BERT on the augmented training set and then evaluate on both the original development set and the corrupted development set.

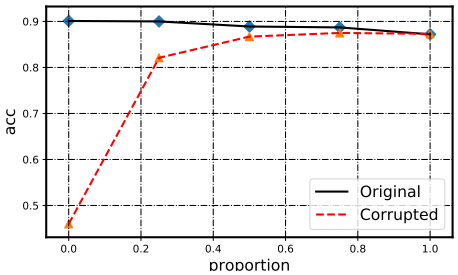

Figure 5: Results of a data augmentation defense. The proportions indicate the amount of adversarial examples augmented to the training set compared to original amount. The dash and solid lines show the accuracy on corrupted and original development set with different proportions respectively.

Results are shown in Figure 5. we find that by adding a small number of adversarial examples, the accuracy is recovered from 46% to 82%. As the proportion of augmented adversarial examples increases, the accuracy continues to increase on the corrupted set, with negligible changes to the original validation accuracy. This fact also demonstrates that our simulated examples are potentially helpful for reducing the effect of grammatical errors.

8 Conclusion

In this paper, we conducted a thorough study to evaluate the robustness of language encoders against grammatical errors. We proposed a novel method to simulating grammatical errors and facilitating our evaluations. We studied three pre-trained language encoders, ELMo, BERT, and RoBERTa and concentrated on three aspects of their abilities against grammatical errors: performance on downstream tasks when confronted with noised texts, ability in identifying errors and capturing the interaction between tokens in the presence of errors. This study shed light on understanding the behaviors of language encoders against grammatical errors and encouraged future work to enhance the robustness of these models.

Acknowledgements We would like to thank the anonymous reviewers for their feedback. This work is supported by NSF Grant #IIS-1927554.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2890–2896.

Antonios Anastasopoulos. 2019. An analysis of source-side grammatical errors in nmt. In *Proc. BlackboxNLP*.

Timothy Baldwin, Trevor Cohn, and Yitong Li. 2017. Robust training under linguistic adversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 21–27.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Steven Bird and Edward Loper. 2004. NLTK: the natural language toolkit. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain, July 21-26, 2004 - Poster and Demonstration*.

Jill Burstein, Christy Doran, and Thamar Solorio, editors. 2019. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does bert look at? an analysis of bert's attention. In *BlackBoxNLP@ACL*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 670–680.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single \$&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2126–2136.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The NUS corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications, BEA@NAACL-HLT 2013, June 13, 2013, Atlanta, Georgia, USA*, pages 22–31.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. 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 (Burstein et al., 2019), pages 4171–4186.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP 2005, Jeju Island, Korea, October 2005, 2005*.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *CoRR*, abs/1901.05287.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3651–3657.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is BERT really robust? natural language attack on text classification and entailment. *CoRR*, abs/1907.11932.

Najoung Kim, Roma Patel, Adam Poliak, Patrick Xia, Alex Wang, Tom McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, Samuel R. Bowman, and Ellie Pavlick. 2019. Probing what different NLP tasks teach machines about function word comprehension. In *Proceedings of the Eighth Joint*

*Conference on Lexical and Computational Semantics, *SEM@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019*, pages 235–249.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *TACL*, 4:521–535.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Alison Lui, Antonios Anastasopoulos, and David Chiang. 2018. Neural machine translation of text from non-native speakers. *CoRR*, abs/1808.06267.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1192–1202.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019.*, pages 7–14.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392.

Marco Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. pages 856–865.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *CoRR*, cs.CL/0306050.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642.

Matthias Sperber, Jan Niehues, and Alex Waibel. 2017. Toward robust neural machine translation for noisy input sequences.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4593–4601.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Alex Wang, Ian F. Tenney, Yada Pruksachatkun, Katherin Yu, Jan Hula, Patrick Xia, Raghu Pappagari, Shuning Jin, R. Thomas McCoy, Roma Patel, Yinghui Huang, Jason Phang, Edouard Grave, Haokun Liu, Najoung Kim, Phu Mon Htut, Thibault F'evry, Berlin Chen, Nikita Nangia, Anhad Mohananey, Katharina Kann, Shikha Bordia, Nicolas

Patry, David Benton, Ellie Pavlick, and Samuel R. Bowman. 2019b. jiant 1.2: A software toolkit for research on general-purpose text understanding models. http://jiant.info/.

Alex Warstadt and Samuel R. Bowman. 2019. Grammatical analysis of pretrained sentence encoders with acceptability judgments. *CoRR*, abs/1901.03438.

Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, Sheng-Fu Wang, Jason Phang, Anhad Mohananey, Phu Mon Htut, Paloma Jeretic, and Samuel R. Bowman. 2019a. Investigating bert's knowledge of language: Five analysis methods with npis. *CoRR*, abs/1909.02597.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R Bowman. 2019b. Blimp: A benchmark of linguistic minimal pairs for english. *arXiv preprint arXiv:1912.00582*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

A Downstream Task Details

We test on four language understanding and a sequence labeling datasets. Statistics of these datasets are listed in Table 6.

MRPC The Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005) is a paraphrase detection task which aims to predict a binary label for whether two sentences are semantically equivalent.

MNLI The Multi-Genre Natural Language Inference Corpus (MNLI) (Williams et al., 2018) is a broad-domain natural language inference task to predict the relation (entailment, contradiction, neutral) between premise and hypothesis. MNLI contains both the matched (in-domain) and mismatched (cross-domain) sections.

QNLI The Question-answering NLI task (QNLI) is recasted from the Stanford Question Answering Dataset (Rajpurkar et al., 2016), which aims to determine whether a context sentence contains the answer to the question (entailment, not entailment).

SST-2 The Stanford Sentiment Treebank two-way class split (SST-2; (Socher et al., 2013)) is a binary classification task which assigns positive or negative labels to movie review sentences.

CoNLL2003 - NER The shared task of CoNLL-2003 Named Entity Recognition (NER) (Sang and Meulder, 2003) is a token level sequence labeling task to recognize four types of named entity: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups.

Dataset	Train	Dev	Avg Len
MRPC	3.7k	409	22.4
MNLI	393k	19k	10.1
QNLI	105k	5.5k	27.6
SST-2	67k	873	19.5
CoNLL2003	15k	3k	14.8

Table 6: Datasets statistics of MRPC, MNLI, QNLI, SST-2, and CoNLL2003. Train and Dev stands for the number of sentences in the train and development set. Avg Len stands for the average sentence length (in token) of the target sentence being attacked.

B Model Details

B.1 Pre-trained Encoder Details

We study BERT (base, uncased), BERT (base, cased) (for NER only), RoBERTa (base), and ELMo. BERT (base) and RoBERTa (base) have the same architecture. Both of them are deep transformer models with 12 layers and 12 attention heads, 768 hidden size in each layer. They contain a learnable output layer for fine-tuning on `[CLS]` or `<s>`. We use PyTorch implement of BERT and RoBERTa from Wolf et al. (2019) and fine-tune them on downstream tasks. For ELMo, we fix ELMo representations as contextual embeddings of tokens and feed them to a two-layer, 1500D BiLSTM with cross-sentence attention mechanism as implemented in *jiant*. (Wang et al., 2019b).

B.2 Training and Fine-tuning Details

For BERT and RoBERTa, we set the maximum input length to be 128, the maximum number of epochs to be 3, and the dropout to be 0.1 for all tasks. We use Adam (Kingma and Ba, 2015) with an initial learning rate of 2e-5, batch size 16 and no warm-up steps for training. For ELMo, we train the BiLSTM using Adam (Kingma and Ba, 2015) with an initial learning rate of 1e-4, batch size 32. We set the dropout to be 0.2, the maximum number of epochs to be 10 and divide the learning rate by 5 when the performance does not improve for 2 epochs.

B.3 Probing model Details

We use a self-attention layer and a linear classifier to compose the probing component in section 5. The self-attention layer takes as input the hidden representations from the fixed layer i of an encoder, denoted as $h = \{h_1^i, h_2^i, ..., h_n^i\}$ and outputs a sentence representation s_i:

$$s_i = \Sigma_{j=1}^{n} \alpha_j h_j^i \quad (1)$$

$$\alpha_j = softmax(v_b^T tanh(W_a h_j^i)) \quad (2)$$

where W_a is a weight matrix and v_b is a vector of parameters. s_i is fed to the classifier to output the probability of the sentence being linguistically acceptable. The self-attention layer has a hidden dim of 100 and 0.1 dropout. The classifier has 1 layer and 0.1 dropout. The probing model is trained with Adam (Kingma and Ba, 2015) using a learning rate of 0.001, batch size of 8 , L_2 weight decay of 0.001 for 10 epochs and early stop patience of 2.

C Attack Algorithms

We conduct three searching algorithms, *greedy search*, *beam search*, *genetic algorithm* in adversarial attacks based on the real errors on NUCLE (Dahlmeier et al., 2013). For *beam search*, we set the beam size as 5. For *genetic algorithm*, we set the population in each generation to be 60 and set the maximum number of generations to be 23% of the corresponding sentence length. For example, if a sentence has 100 tokens, the genetic algorithm will iterate for at most 23 iterations. Algorithm 1, 2 and 3 are detailed descriptions of greedy attack, beam search attack, and genetic algorithm attack, respectively.

Algorithm 1 Greedy attack

10cm

Input: Original sentence $X_{ori} = \{w_1, w_2, ..., w_n\}$, ground truth prediction Y_{ori}, target model F, all confusion sets P, budget b.

Output: Adversarial example X_{adv}.

Initialization: $X_{adv} \leftarrow X_{ori}$
for each w_i in X_{ori} **do**
    Delete w_i and compute the drop of likelihood on Y_{ori} to obtain the importance score of w_i, denoted as S_{w_i}.
    Apply all substitutions of P to w_i. Obtain the operation pool of w_i, denoted as W_i^{sub}.
end for

Get the index list of X_{ori} according to the decrease order of token importance: $I \leftarrow argsort_{w_i \in X_{ori}}(S_{w_i})$
for each i in I **do**
    $p_{ori} \leftarrow F(X_{adv})|_{Y=Y_{ori}}$
    for each $w^{'}$ in W_i^{sub} **do**
        Substitute w_i with $w^{'}$ in X_{adv} (or swap their positions),
        $Y_{adv} \leftarrow argmaxF(X_{adv})$, $p_{adv} \leftarrow F(X_{adv})|_{Y=Y_{ori}}$
        if not $Y_{ori} = Y_{adv}$ **then return** X_{adv}
        else
            if $p_{adv} < p_{ori}$ **then**
                $w_{select} \leftarrow w^{'}$, $p_{ori} \leftarrow p_{adv}$
            end if
        end if
    end for
    if the number of iterations exceed b **then return** X_{ori}
    end if
    Substitute w_i with w_{select} in X_{adv},
end for
return X_{ori}

D Probing Model Ability in Identifying Errors

D.1 The Sentence-level Binary Classification Task

Table 7 shows complete results for probing individual layers of ELMo, BERT, and RoBERTa across eight error types in the sentence-level binary classification task. We fix the parameters of pre-trained encoders and train a self-attention classifier for each layer to judge the binary linguistic acceptability of a sentence. We find that layer 1 of ELMo, middle layers of BERT, and top layers of RoBERTa perform the best in this evaluation.

D.2 The Token-level Error Locating Task

Table 8 shows complete results for probing individual layers of ELMo, BERT, and RoBERTa across eight error types in the token-level. We first fix the parameters of pre-trained encoders and train a self-attention classifier for each layer to judge the binary linguistic acceptability of a sentence. Then, we extract the two positions with the highest attention weights of self-attention layers and see if error tokens are included.

E Case Study of Locating Error Positions

We show some examples of the token-level evaluation in section 5. We randomly select one example for each error type and visualize the attention weights of the self-attention layer upon different layers of BERT. A deeper purple under each token means the self-attention layer is putting more attention on this token.

F The Token-level Evaluation on Attention Heads of BERT

As mentioned in section 5. We also conduct the same token-level probing to 144 attention heads of BERT. In this experiment, the parameters in BERT are completely frozen. We observe that even in this unsupervised manner, some attention heads are still capable of precisely locating error positions. Middle layers of BERT perform the best. We further observe that some attention heads might be associated with specific types of errors. For example, head 2 in layer 9 and head 6 in layer 10 are good at capturing `SVA` and `Vform`. Both of these two errors are related to verbs.

Algorithm 2 Beam search attack

Input: Original sentence $X_{ori} = \{w_1, w_2, ..., w_n\}$, ground truth prediction Y_{ori}, target model F, all confusion sets P, budget b, beam size bm.
Output: Adversarial example X_{adv}.

Initialization: $bestBeam \leftarrow$ copy X_{ori} for bm times.
for each w_i in X_{ori} **do**
    Delete w_i and compute the drop of likelihood on Y_{ori} to obtain the importance score of w_i, denoted as S_{w_i}.
    Apply all substitutions of P to w_i. Obtain the operation pool of w_i, denoted as W_i^{sub}.
end for

Get the index list of X_{ori} according to the decrease order of token importance: $I \leftarrow argsort_{w_i \in X_{ori}}(S_{w_i})$
for each $w^{'}$ in $W_{I[0]}^{sub}$ **do**
    Substitute w_i with $w^{'}$ in X_{ori} (or swap their positions)
    $Y_{adv} \leftarrow argmaxF(X_{ori})$, $p_{adv} \leftarrow F(X_{ori})|_{Y=Y_{ori}}$
    if not $Y_{ori} = Y_{adv}$ **then return** X_{ori}
    else
        $topBeam \leftarrow$ Record top-bm examples with the lowest p_{adv}
    end if
end for

$bestBeam \leftarrow topBeam$
for each i in $I/I[0]$ **do**
    $p_{ori} \leftarrow F(X_{adv})|_{Y=Y_{ori}}$
    $oplist \leftarrow \{\}$
    for each X_{beam} in $bestBeam$ **do**
        for each $w^{'}$ in W_i^{sub} **do**
            Substitute w_i with $w^{'}$ in X_{beam} (or swap their positions)
            $Y_{adv} \leftarrow argmaxF(X_{beam})$, $p_{adv} \leftarrow F(X_{beam})|_{Y=Y_{ori}}$
            if not $Y_{ori} = Y_{adv}$ **then return** X_{beam}
            else
                Add $op \leftarrow (w^{'}, p_{adv}, X_{beam})$ to $oplist$
            end if
        end for
    end for
    if number of iterations exceed b **then return** X_{ori}
    end if
    Select the top-bm ops in $oplist$ with lowest $op.p_{adv}$. Update $bestBeam$ with each $op.X_{beam}$.
end for
return X_{ori}

Algorithm 3 Genetic attack

Input: Original sentence $X_{ori} = \{w_1, w_2, ..., w_n\}$, ground truth prediction Y_{ori}, target model F, all confusion sets P, budget b, population size ps, generation size G.
Output: Adversarial example X_{adv}.

Initialize the first generation with empty set: $P^0 \leftarrow \emptyset$.
for each w_i in X_{ori} **do**
    Apply all substitutions of P to w_i. Obtain the operation pool of w_i, denoted as W_i^{sub}.
end for
for $i = 1, 2, 3, ..., ps$ **do**
    Randomly select a position j and an operation from W_j^{sub} to modify X_{ori}. Then add to P^0.
end for

for $g = 1, 2, 3, ..., G-1$ **do**
    for $i = 1, 2, 3, ..., ps$ **do**
        $Y_{adv} \leftarrow argmaxF(P_i^{g-1})$, $p_{adv} \leftarrow F(P_i^{g-1})|_{Y=Y_{ori}}$
        if not $Y_{adv} = Y_{ori}$ **then return** P_i^{g-1}
        else
            $X_{elite} \leftarrow argmin(p_{adv})$
            $P_1^g \leftarrow \{X_{elite}\}$
            $prob \leftarrow$ Normalize sample probability with $F(P_i^{g-1})$
            for $i = 2, 3, ..., ps$ **do**
                Sample parent1 from P^{g-1} with probs $prob$
                Sample parent2 from P^{g-1} with probs $prob$
                $child \leftarrow$ Crossover(parent1, parent2)
                $child_{mut} \leftarrow$ Randomly select a position and an operation from W_j^{sub} to modify $child$
                $P_i^g \leftarrow child_{mut}$
            end for
        end if
    end for
end for
return X_{ori}

	Prep	Artordet	Vform	Nn	Wchoice	Trans	SVA	Worder
ELMo, layer 0	62.6	65.0	69.6	67.7	74.5	67.5	72.1	47.6
ELMo, layer 1	**90.6**	**84.7**	**87.2**	**82.9**	**83.9**	**80.6**	**93.1**	**71.2**
ELMo, layer 2	84.7	77.0	79.4	79.7	82.6	74.4	89.9	68.5
BERT, layer 0	62.5	60.8	67.4	64.6	73.9	69.5	70.3	48.2
BERT, layer 1	68.0	63.4	69.3	70.3	75.0	71.5	78.4	52.2
BERT, layer 2	74.4	67.0	75.3	74.8	76.7	73.1	84.4	62.0
BERT, layer 3	80.5	75.0	83.4	73.7	78.5	76.3	89.2	69.8
BERT, layer 4	82.7	80.7	83.6	77.7	82.6	79.6	90.6	72.4
BERT, layer 5	85.2	83.8	85.4	84.3	84.5	81.8	91.7	71.9
BERT, layer 6	88.2	86.6	85.8	86.7	84.5	82.6	90.9	73.4
BERT, layer 7	91.3	88.1	90.2	86.5	**86.9**	83.9	**95.3**	73.4
BERT, layer 8	**92.5**	**88.3**	**91.4**	**88.4**	86.3	**85.5**	94.5	**73.8**
BERT, layer 9	91.4	86.3	89.9	87.4	85.6	84.9	94.4	72.4
BERT, layer 10	90.8	87.4	88.2	87.0	86.1	84.8	94.9	71.8
BERT, layer 11	90.0	84.9	88.1	86.6	85.6	84.3	94.2	69.5
BERT, layer 12	88.4	85.6	88.1	84.3	84.0	82.6	93.3	68.1
RoBERTa, layer 0	61.9	65.9	69.7	67.1	75.1	69.1	68.3	50.9
RoBERTa, layer 1	78.3	74.7	84.6	77.6	80.2	75.9	88.4	67.8
RoBERTa, layer 2	85.2	79.4	88.7	83.0	83.3	78.8	90.9	71.8
RoBERTa, layer 3	89.3	85.7	90.6	86.9	87.0	84.1	94.3	72.6
RoBERTa, layer 4	90.2	88.7	91.8	88.7	86.2	86.4	94.5	74.5
RoBERTa, layer 5	91.4	89.1	92.9	90.5	89.0	87.1	95.5	74.5
RoBERTa, layer 6	93.4	91.3	91.9	91.4	88.9	86.8	95.0	75.3
RoBERTa, layer 7	93.9	90.5	91.8	90.4	88.2	86.9	94.6	74.7
RoBERTa, layer 8	93.9	91.1	**93.4**	92.3	88.0	87.2	94.4	75.9
RoBERTa, layer 9	94.3	90.6	92.5	92.1	89.4	88.0	**95.7**	74.7
RoBERTa, layer 10	94.4	**92.0**	93.3	**92.3**	**89.9**	88.1	95.0	75.1
RoBERTa, layer 11	**95.3**	91.5	93.3	89.4	88.8	**88.2**	95.2	**76.0**
RoBERTa, layer 12	94.5	91.1	92.7	88.3	87.3	87.9	95.3	74.8

Table 7: Results of the accuracy on the binary linguistic acceptability probing task for individual layers of ELMo, BERT, and RoBERTa.

	Prep	Artordet	Vform	Nn	Wchoice	Trans	SVA	Worder
ELMo, layer 0	23.2	14.3	22.3	9.8	21.8	10.2	18.4	29.6
ELMo, layer 1	56.5	**42.6**	51.8	82.0	72.0	**69.4**	30.6	55.1
ELMo, layer 2	**68.0**	34.2	**55.4**	**85.9**	**73.0**	42.8	**49.2**	**62.7**
BERT, layer 0	24.1	39.1	66.7	58.7	62.3	56.4	63.6	17.5
BERT, layer 1	56.6	33.9	66.9	59.3	69.4	71.1	54.4	13.1
BERT, layer 2	58.7	27.4	75.8	58.4	76.3	83.3	60.0	34.1
BERT, layer 3	64.5	55.2	56.2	62.4	79.3	83.0	64.2	67.8
BERT, layer 4	68.9	54.1	69.2	62.9	81.7	66.0	67.3	59.7
BERT, layer 5	67.4	52.4	76.9	60.8	83.8	80.7	62.2	62.3
BERT, layer 6	68.2	51.5	76.5	58.7	84.9	**83.9**	71.7	66.9
BERT, layer 7	70.4	**52.3**	93.0	61.8	82.8	81.9	61.3	61.2
BERT, layer 8	69.9	51.7	93.0	65.4	80.2	80.2	60.9	**63.9**
BERT, layer 9	**71.7**	48.0	91.6	**85.3**	**84.9**	79.6	59.6	62.2
BERT, layer 10	70.7	50.4	90.5	80.5	82.3	78.2	92.4	58.7
BERT, layer 11	70.1	49.2	**96.3**	80.5	81.0	80.7	90.5	60.3
BERT, layer 12	71.4	50.5	86.7	79.8	79.1	81.6	**93.2**	58.8
RoBERTa, layer 0	44.8	26.5	74.8	62.8	71.3	71.1	61.7	14.3
RoBERTa, layer 1	68.3	12.1	90.7	62.5	80.9	75.9	93.5	48.9
RoBERTa, layer 2	69.9	35.3	71.0	61.9	83.9	84.1	60.5	58.2
RoBERTa, layer 3	71.9	54.4	92.2	60.7	85.5	84.4	**96.2**	59.3
RoBERTa, layer 4	71.2	48.9	92.0	83.3	85.6	**85.3**	95.9	60.8
RoBERTa, layer 5	**71.9**	**53.6**	92.5	84.9	**88.5**	83.9	95.3	61.2
RoBERTa, layer 6	70.2	52.9	92.5	87.0	87.3	83.9	95.7	59.0
RoBERTa, layer 7	70.6	50.6	92.1	87.8	87.2	83.9	94.8	58.4
RoBERTa, layer 8	71.6	51.5	92.2	**89.5**	87.0	79.6	95.2	58.8
RoBERTa, layer 9	71.3	53.2	91.9	87.7	86.7	81.3	95.8	61.1
RoBERTa, layer 10	69.6	50.3	**92.8**	86.8	87.1	78.8	96.0	**64.2**
RoBERTa, layer 11	69.3	49.6	92.7	88.4	86.5	75.6	95.5	62.0
RoBERTa, layer 12	69.6	48.9	90.1	86.8	84.9	79.6	94.1	62.8

Table 8: Results of the accuracy on locating error positions for individual layers of ELMo, BERT, and RoBERTa.

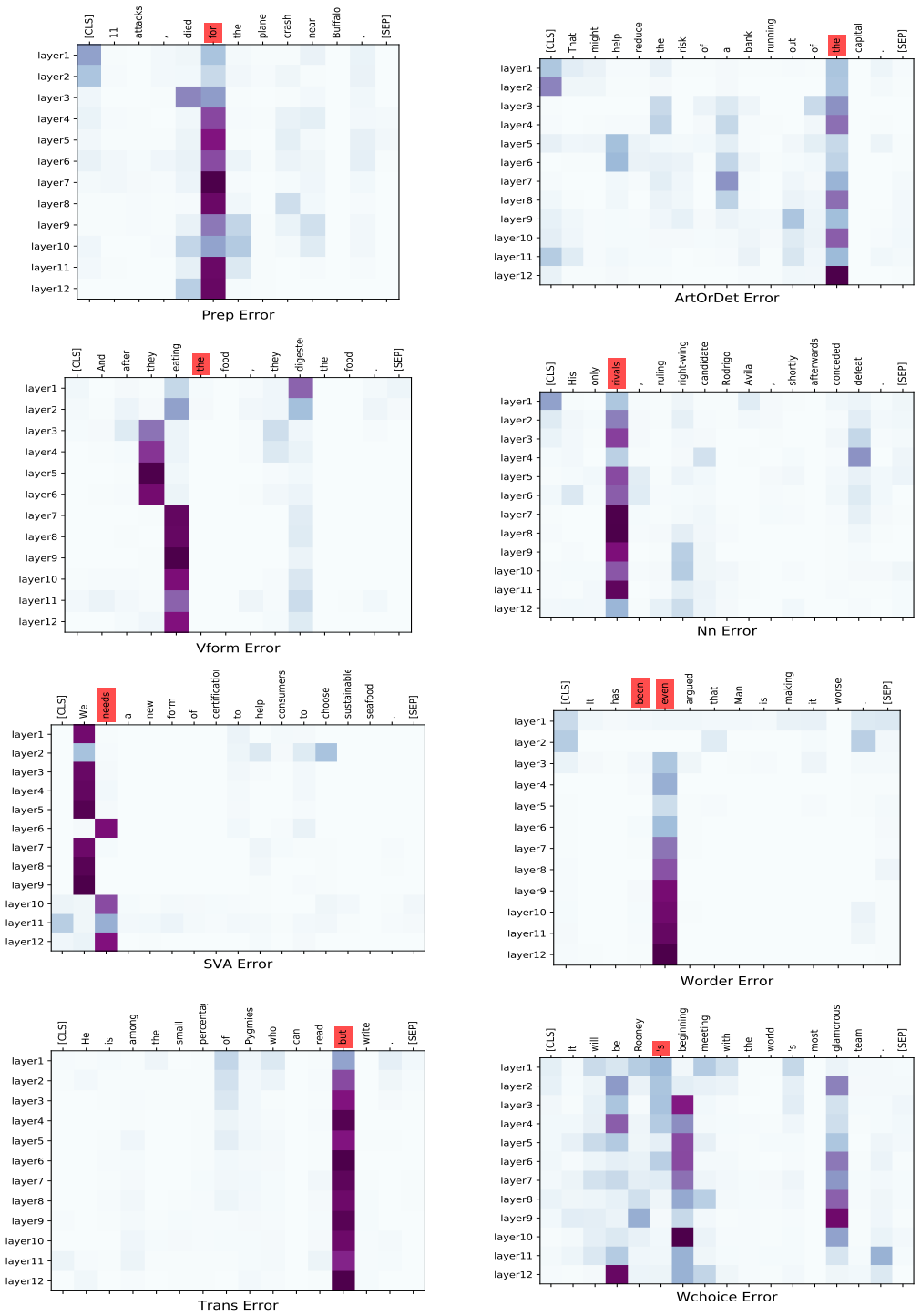

Figure 6: Visualization of attention weights of self-attention layers. A figure represents a sentence with a specific error type. Errors in a sentence are highlighted in red. Each column represents one layer of BERT that the self-attention layer is build upon.

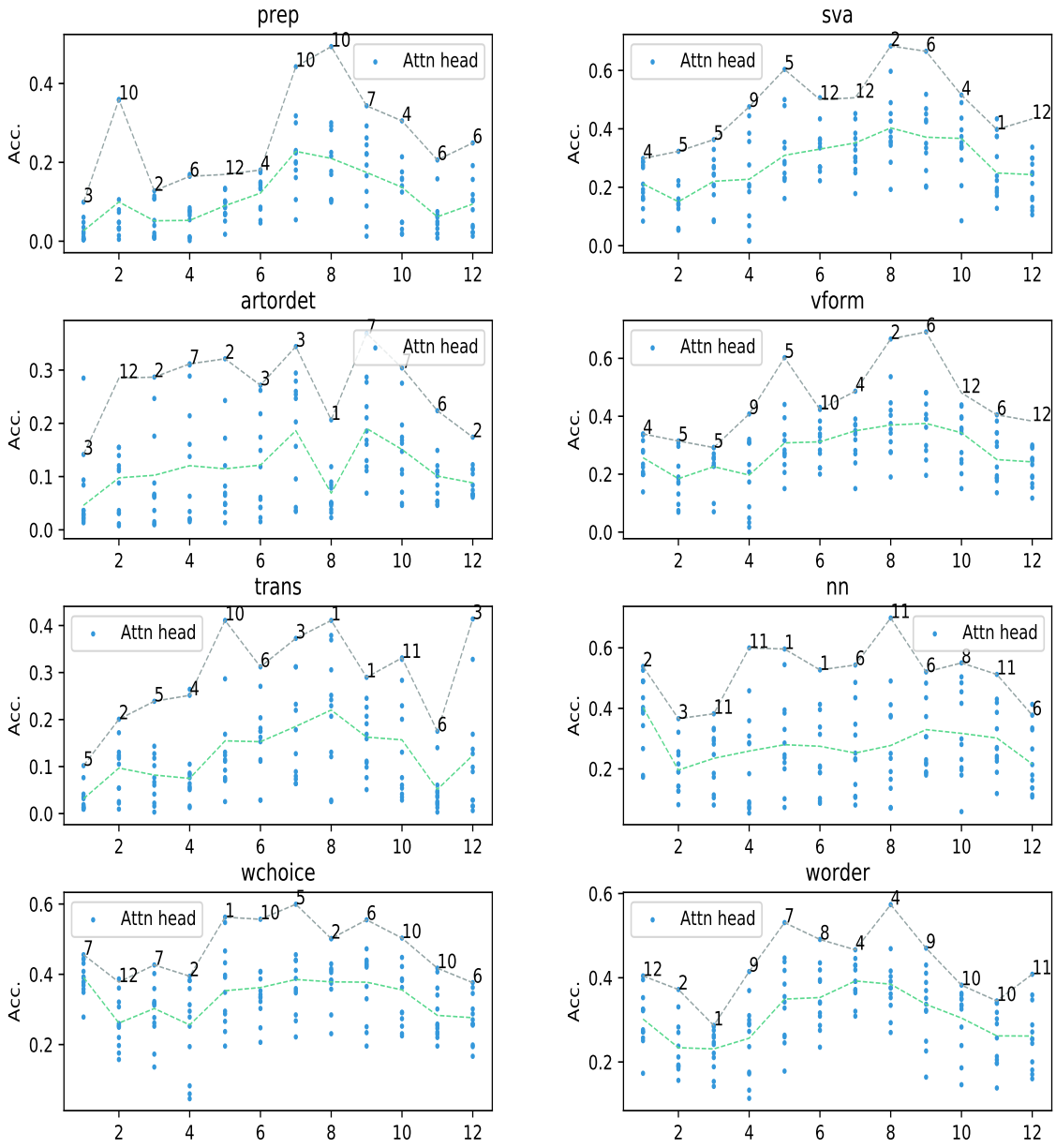

Figure 7: Visualization for each attention head of BERT for locating each type of error. A point in the figure represents the performance of an attention head. The grey line on the top represents the best performing head in each layer (annotated with its number). The green line in the middle represents the average performance of all heads in this layer.