On the Robustness of Language Encoders against Grammatical Errors

**Fan Yin**1**, Quanyu Long**2**, Tao Meng**3**, and Kai-Wei Chang**3 1Peking University

2Shanghai Jiao Tong University 3University of California, Los Angeles 1600012975@pku.edu.cn; oscar.long@sjtu.edu.cn; tmeng@cs.ucla.edu; [kwchang@cs.ucla.edu](mailto:kwchang@cs.ucla.edu)

# Abstract

We conduct a thorough study to diagnose the behaviors of pre-trained language en- coders (ELMo, BERT, and RoBERTa) when confronted with natural grammatical errors. Speciﬁcally, we collect real grammatical er- rors from non-native speakers and conduct ad- versarial attacks to simulate these errors on clean text data. We use this approach to facil- itate debugging models on downstream appli- cations. Results conﬁrm that the performance of all tested models is affected but the degree of impact varies. To interpret model behav- iors, we further design a linguistic acceptabil- ity 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 predic- tion of sentence correctness are able to locate error positions. We also design a cloze test for BERT and discover that BERT captures the in- teraction between errors and speciﬁc tokens in context. Our results shed light on understand- ing the robustness and behaviors of language encoders against grammatical errors.

# Introduction

Pre-trained language encoders have achieved great success in facilitating various downstream natu- ral language processing (NLP) tasks ([Peters et al.](#_bookmark36), [2018](#_bookmark36); [Devlin et al.](#_bookmark24), [2019](#_bookmark24); [Liu et al.](#_bookmark33), [2019b](#_bookmark33)). How- ever, they usually assume training and test cor- pora are clean and it is unclear how the models behave when confronted with noisy input. Gram- matical error is an important type of noise since it naturally and frequently occurs in natural lan- guage, especially in spoken and written materials from non-native speakers. Dealing with such a noise reﬂects model robustness in representing lan- guage and grammatical knowledge. It would also have a positive social impact if language encoders

can model texts from non-native speakers appropri- ately.

Recent work on evaluating model’s behaviors against grammatical errors employs various meth- ods, including (1) manually constructing mini- mal edited pairs on speciﬁc linguistic phenom- ena ([Marvin and Linzen](#_bookmark35), [2018](#_bookmark35); [Goldberg](#_bookmark26), [2019](#_bookmark26); [Warstadt et al.](#_bookmark50), [2019a](#_bookmark50),[b](#_bookmark51)); (2) labeling or creating acceptability judgment resources ([Linzen et al.](#_bookmark31), [2016](#_bookmark31); [Warstadt and Bowman](#_bookmark49), [2019](#_bookmark49); [Warstadt et al.](#_bookmark50), [2019a](#_bookmark50)); and (3) simulating noises for a speciﬁc NLP task such as neural machine translation ([Lui](#_bookmark34) [et al.](#_bookmark34), [2018](#_bookmark34); [Anastasopoulos](#_bookmark13), [2019](#_bookmark13)), sentiment clas- siﬁcation ([Baldwin et al.](#_bookmark14), [2017](#_bookmark14)). These studies either focus on speciﬁc phenomena and mainly conduct experiments on designated corpora or rely heavily on human annotations and expert knowl- edge in linguistics. In contrast, our work automat- ically simulates natural occurring data and vari- ous types of grammatical errors and systematically analyzes how these noises affect downstream ap- plications. This holds more practical signiﬁcance to understand the robustness of several language encoders against grammatical errors.

Speciﬁcally, we ﬁrst propose an effective ap- proach to simulating diverse grammatical errors, which applies black-box adversarial attack algo- rithms based on real errors observed on NUS Cor- pus of Learner English (NUCLE) ([Dahlmeier et al.](#_bookmark22), [2013](#_bookmark22)), a grammatical error correction benchmark. This approach transforms clean corpora into cor- rupted ones and facilitates debugging language en- coders 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 identify- ing grammatical errors by probing individual layers of pre-trained encoders through a linguistic accept- ability task. We construct separate datasets for eight error types. Then, we freeze encoder layers

3386

*Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3386–3403 July 5 - 10, 2020. *O*c 2020 Association for Computational Linguistics

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 sim- ple classiﬁer behaves well on a designated type of error, then the encoder layer is likely to contain knowledge of that error ([Conneau et al.](#_bookmark20), [2017](#_bookmark20); [Adi](#_bookmark11) [et al.](#_bookmark11), [2017](#_bookmark11)).

Finally, we investigate how models capture the interaction between grammatical errors and con- texts. 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 ro- bustness 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 exist- ing language encoders against grammatical errors varies; (2) the contextual layers of lan- guage 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 avail- able at: [https://github.com/uclanlp/](https://github.com/uclanlp/ProbeGrammarRobustness) [ProbeGrammarRobustness](https://github.com/uclanlp/ProbeGrammarRobustness)

# Related Work

**Probing Pre-trained Language Encoders** The recent success of pre-trained language encoders across a diverse set of downstream tasks has stim- ulated 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.](#_bookmark44) ([2019a](#_bookmark44)), these stud- ies can be roughly divided into two categories: (1) designing controlled tasks to probe whether a spe- ciﬁc linguistic phenomenon is captured by models ([Conneau et al.](#_bookmark21), [2018](#_bookmark21); [Peters et al.](#_bookmark37), [2019](#_bookmark37); [Tenney](#_bookmark45) [et al.](#_bookmark45), [2019b](#_bookmark45); [Liu et al.](#_bookmark32), [2019a](#_bookmark32); [Kim et al.](#_bookmark29), [2019](#_bookmark29)), or

(2) decomposing the model structure and exploring what linguistic property is encoded ([Tenney et al.](#_bookmark44),

[2019a](#_bookmark44); [Jawahar et al.](#_bookmark27), [2019](#_bookmark27); [Clark et al.](#_bookmark19), [2019](#_bookmark19)). However, these studies do not analyze how gram- matical errors affect model behaviors.

Our work is related to studies on analyzing mod- els with manually created noise. For example, [Linzen et al.](#_bookmark31) ([2016](#_bookmark31)) evaluate whether LSTMs cap- ture the hierarchical structure of language by using verbal inﬂection to violate subject-verb agreement. [Marvin and Linzen](#_bookmark35) ([2018](#_bookmark35)) present a new dataset consisting of minimal edited pairs with the oppo- site linguistic acceptability on three speciﬁc lin- guistic phenomena and use it to evaluate RNN’s syntactic ability. [Goldberg](#_bookmark26) ([2019](#_bookmark26)) adjusts previous method to evaluate BERT. [Warstadt et al.](#_bookmark50) ([2019a](#_bookmark50)) further compare ﬁve analysis methods under a sin- gle phenomenon. Despite the diversity in methodol- ogy, 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.](#_bookmark51), [2019b](#_bookmark51)) facilitates diagnosing language models by creat- ing linguistic minimal pairs datasets for 67 isolate grammatical paradigms in English using linguist- crafted templates. In contrast, we do not rely heav- ily 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.](#_bookmark31), [2016](#_bookmark31); [Marvin](#_bookmark35) [and Linzen](#_bookmark35), [2018](#_bookmark35); [Goldberg](#_bookmark26), [2019](#_bookmark26); [Warstadt et al.](#_bookmark50), [2019a](#_bookmark50)). Others have introduced various methods to generate synthetic errors on clean downstream datasets, in particular, machine translation corpora. [Belinkov and Bisk](#_bookmark15) ([2018](#_bookmark15)); [Anastasopoulos](#_bookmark13) ([2019](#_bookmark13)) demonstrate that synthetic grammatical errors in- duced by character manipulation and word substitu- tion can degrade the performance of NMT systems. [Baldwin et al.](#_bookmark14) ([2017](#_bookmark14)) augment original sentiment classiﬁcation datasets with syntactically (reorder- ing) and semantically (word substitution) noisy sentences and achieve higher performance. Our method is partly inspired by [Lui et al.](#_bookmark34) ([2018](#_bookmark34)), who synthesize semi-natural ungrammatical sentences by maintaining confusion matrices for ﬁve simple error types.

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

for debugging NLP models ([Ribeiro et al.](#_bookmark40), [2018](#_bookmark40); [Jin et al.](#_bookmark28), [2019](#_bookmark28); [Alzantot et al.](#_bookmark12), [2018](#_bookmark12); [Burstein et al.](#_bookmark17), [2019](#_bookmark17)). These methods create a more challenging scenario for target models compared to the above data generation procedure. Our proposed simu- lation beneﬁts from both adversarial attack algo- rithms and semi-natural grammatical errors.

# Method

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

## Grammatical Error Simulation

Most downstream datasets contain only clean and grammatical sentences. Despite that recent lan- guage 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.](#_bookmark34), [2018](#_bookmark34); [Sperber](#_bookmark43) [et al.](#_bookmark43), [2017](#_bookmark43)) and apply two ways to introduce er- rors to clean corpora: (1) we sample errors based on the frequency distribution of NUCLE and intro- duce them to plausible positions; (2) inspired by the literature of adversarial attacks ([Ribeiro et al.](#_bookmark40), [2018](#_bookmark40); [Jin et al.](#_bookmark28), [2019](#_bookmark28); [Alzantot et al.](#_bookmark12), [2018](#_bookmark12)), we conduct search algorithms to introduce grammati- cal 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.](#_bookmark22), [2013](#_bookmark22)). NUCLE is con- structed with naturally occurring data (student es- says at NUS) annotated with error tags. Each un- grammatical 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 **grow- ing** up?

*×*

* + 1. 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 prepo- sition errors), ArtOrDet (indicating article or de- terminer 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](#_bookmark1).

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 *wij* between token *ti* and *tj* in the same set to indicate the probability that *ti* will be replaced by *tj*. 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 *ti* in the set, we compute *wij* based on how many times *ti* is replaced by *tj* in minimal edited pairs on NUCLE.

Notice that we add a special token ø to repre- sent 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 ad- jective, participle or modal. For Vform, we use NLTK ([Bird and Loper](#_bookmark16), [2004](#_bookmark16)) to extract present, past, progressive, and perfect tense of a verb and add to the set. For Wchoice, we select ten syn- onyms of a target word from WordNet. The substi- tution weight is set to be uniform for both Vform and Wchoice.

**Grammatical Error Introduction** We intro- duce errors in two ways. The ﬁrst is called *proba- bilistic transformation*. Similar to [Lui et al.](#_bookmark34) ([2018](#_bookmark34)), we ﬁrst obtain the parse tree of the target sentence using the Berkeley syntactic parser ([Petrov et al.](#_bookmark38), [2006](#_bookmark38)). 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 rep- resents the average case. To debug and analyze the robustness of language encoders, we consider an- other more challenging setting – *worst-case trans- formation*, where we leverage search algorithms

## Error type Error Description Confusion Set

ArtOrDet Article/determiner errors *{* a, an, the, ø*}*

*{* on, in, at, from, for, under, over, with, into,

Prep Preposition errors

Trans Link words/phrase errors

during, until, against, among, throughout, to, by, about, like, before, across, behind, but, out, up, after, since, down, off, of, ø

and, but, so, however, as, that, thus, also, be- cause, therefore, if, although, which, where, moreover, besides, of, ø*}*

*{*

*}*

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 op- eration set for each token in a sentence by consider- ing all possible substitutions based on all confusion sets. Note that some confusion sets are not applica- ble, 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 ap- ply 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 predic- tion when it is deleted. The larger the decrease is, the more important the token is. After compar- ing 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 associ- ated with that token and then practice the operation that degrades the likelihood of the model predic- tion 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.](#_bookmark12) ([2018](#_bookmark12)) for details of the genetic algorithm in adversarial attack. Com- prehensive descriptions of all methods are found in Appendix [C](#_bookmark58).

## Target Models

We evaluate the following three pre-trained lan- guage encoders. Detailed descriptions of models and training settings are in Appendix [B](#_bookmark55).

**ELMo** ([Peters et al.](#_bookmark36), [2018](#_bookmark36)) is a three-layer LSTM-based model pre-trained on the bidirectional language modeling task on 1B Word Benchmark ([Chelba et al.](#_bookmark18), [2014](#_bookmark18)). We ﬁx ELMo as a contextual embedding and add two layers of BiLSTM with attention mechanism on top of it.

**BERT** ([Devlin et al.](#_bookmark24), [2019](#_bookmark24)) is a transformer- based ([Vaswani et al.](#_bookmark46), [2017](#_bookmark46)) model pre-trained on masked language modeling and next sentence pre- diction 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 per- form task-speciﬁc ﬁne-tuning.

**RoBERTa** ([Liu et al.](#_bookmark33), [2019b](#_bookmark33)) is a robustly pre- trained BERT model using larger pre-training data (160GB in total), longer pre-training time, the dy- namic masking strategy and other optimized pre-

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

## 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](#_bookmark0), we are able to perform evaluation on existing bench- mark 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 remain- ing 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 sen- tence 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 to- ken, we quantify how the probability of predicting a single masked token in the rest of the sentence affected by this grammatical error. This method an- alyzes how error token affects clean context, which is complementary to [Goldberg](#_bookmark26) ([2019](#_bookmark26)) who focuses on SVA error and discusses how clean contexts inﬂuence the prediction of the masked error token.

# 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.](#_bookmark20), [2017](#_bookmark20)).

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](#_bookmark25), [2005](#_bookmark25)), MNLI ([Williams et al.](#_bookmark52), [2018](#_bookmark52)), QNLI ([Rajpurkar](#_bookmark39) [et al.](#_bookmark39), [2016](#_bookmark39)), and SST-2 ([Socher et al.](#_bookmark42), [2013](#_bookmark42)) from GLUE ([Wang et al.](#_bookmark47), [2019a](#_bookmark47)) and a sequence tagging benchmark: CoNLL-2013 for NER. Detailed de- scriptions of these corpora are in Appendix [A](#_bookmark54). 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 max- imum percentage of allowed modiﬁcations in a sentence to be 15% of tokens, which is a reason- able rate according to the statistics estimated from the real data. As shown in Table [3](#_bookmark3), 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](#_bookmark2) presents the test performance of four target models on the standard development set of each task. Table [3](#_bookmark3) 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](#_bookmark3), with the *probabilistic trans- formation*, the attack success rates fall between 2% (RoBERTa, QNLI) and 10% (ELMo, MRPC). With

the *worst-case transformation*, we obtain the high- est attacked rate of 81.1% (ELMo, genetic algo- rithm, 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 ob- serve that the impact of grammatical errors varies among models. Speciﬁcally, RoBERTa exhibits a strong robustness against the impact of gram- matical errors, with consistently lower attack suc- cess rates (20.28% on average) and F1 score de- creases (17.50% on average) across all tasks, es- pecially on MRPC and MNLI. On the other hand, BERT, ELMo, and InferSent experience an aver- age attack rate of 26.03%, 33.06%, 36.07% re- spectively on NLU tasks. Given the differences in pre-training strategies, we speculate that pre-

training with more data might beneﬁt model ro- bustness against noised data. This speculation is consistent with ([Warstadt et al.](#_bookmark51), [2019b](#_bookmark51)), where the authors also give a lightweight demonstration on LSTM and Transformer-XL ([Dai et al.](#_bookmark23), [2019](#_bookmark23)) with varying training data. We leave a further explo- ration 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 at- tack 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 differ- ent 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

1. 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 grammati- cal 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 ana- lyze BERT and RoBERTa on the development sets of MRPC, MNLI-m, and SST-2 as shown in Table

**Sentence-level Acc Token-level Acc**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  | InferSent  ELMo |  |
|  |  |  |  |  | BERT |  |
|  |  |  |  |  | RoBERTa |  |

0.8

Attack success rate

0.7

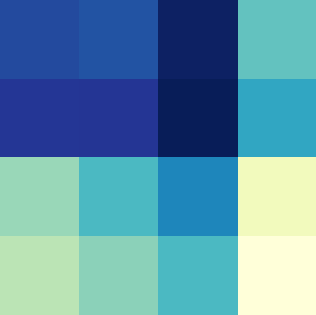
0.6

0.5

0.4

0.3

**layer 12**

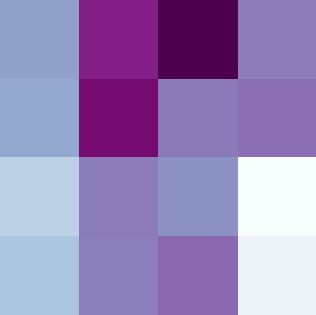
**layer 8**

|  |  |  |  |
| --- | --- | --- | --- |
| 85.6 | 84.3 | 93.3 | 68.1 |
| 88.3 | 88.4 | 94.5 | 73.8 |
| 63.4 | 70.3 | 78.4 | 52.2 |
| 60.8 | 64.6 | 70.3 | 48.2 |

**layer 1**

**layer 0**

**layer 12**

**layer 9**

|  |  |  |  |
| --- | --- | --- | --- |
| 50.5 | 79.8 | 93.2 | 58.8 |
| 48.0 | 85.3 | 59.6 | 62.2 |
| 33.9 | 59.3 | 54.4 | 13.1 |
| 39.1 | 58.7 | 63.6 | 17.5 |

**layer 1**

**layer 0**

0.12 0.14 0.16 0.18 0.20 0.22

Perturbed rate

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

1. 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 differ- ent 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 percent- age of modiﬁed tokens under 20%. We evaluate all models on the development set of MNLI-m. Re- sults are shown in Fig [1](#_bookmark5). 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 begin- ning and RoBERTa grows slower when reaching the end. The average attack success rate comes to 70% when the error rate is around 20%.

# 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 grammat- ical correctness of a sentence. We further study whether the model can precisely locate error posi- tions, which reﬂects the token-level ability.

**Data** We construct separate datasets for each spe- ciﬁc 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.](#_bookmark18), [2014](#_bookmark18)). Then, we introduce the target error type to half of these sentences using *proba- bilistic transformation* and keep the error rate over each dataset around 3% (resulting in one or two



50 60 70 80 90 20 30 40 50 60 70 80 90

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 repre- sents 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](#_bookmark60)

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](#_bookmark56). 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 infor- mation for the classiﬁer to identify error locations. For comparisons, we also evaluate the input em- bedding 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 re- sults of four layers of BERT on four error types, ArtOrDet, Nn, SVA, and Worder in Fig [2](#_bookmark6). Complete results of all layers and other error types are in Appendix [D](#_bookmark60). 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

0.5

0.4

0.3

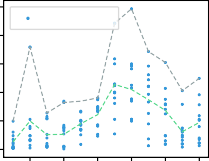
Acc.

0.2

0.1

0.0

**prep**



10

Attn head10

10

7

4

6

6

6 12 4

2

3

2 4 6 8 10 12

Acc.

0.6

0.4

0.2

0.0

**sva**



2 6

5

12 12

4

9

12

1

5

4 5

Attn head

2 4 6 8 10 12

Prep Art Wci Tras Nn SVA

Vform

-6 -5 -4 -3 -2 -1 1 2 3 4 5 6

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.

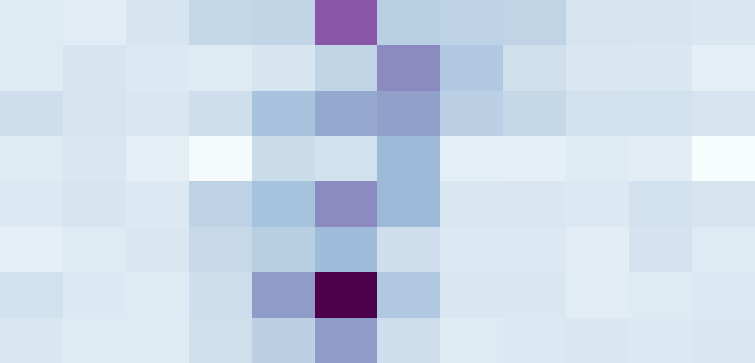
prediction of whether a sentence is acceptable, the mean accuracy of classiﬁers upon the best layers of BERT, ELMo, and RoBERTa are 79.3%, 63.3%,

and 80.3%, compared to 48.6%, 18.7%, and 53.4% of input embedding layers. The two facts indi- cate 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](#_bookmark6). 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](#_bookmark7). We ﬁnd that even in a completely unsu- pervised manner, some attention heads results for 50%-60% accuracy in locating errors. Consistent with self-attention layers, attention heads from mid- dle layers perform the best. See Appendix [F](#_bookmark62) for all error types.

Due to space limit, we present results of RoBERTa and ELMo in Appendix [D](#_bookmark60) and summa- rize 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 de- pendency errors. On the other hand, the ﬁrst layer of ELMo consistently gives the highest sentence- level classiﬁcation accuracy. But its best perform- ing 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 ex- hibits strong ability in locating Nn and outperforms BERT in accuracy. This is surprising given the fact that Nn is not obvious with character embeddings

Vt

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

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.00 | -0.00 | 0.01 | 0.02 | 0.02 | 0.09 | 0.02 | 0.02 | 0.02 | 0.01 | 0.01 | 0.00 |
| 0.00 | 0.01 | 0.00 | 0.00 | 0.01 | 0.02 | 0.06 | 0.03 | 0.01 | 0.00 | 0.00 | -0.00 |
| 0.01 | 0.01 | 0.00 | 0.01 | 0.03 | 0.05 | 0.05 | 0.02 | 0.02 | 0.01 | 0.01 | 0.01 |
| 0.00 | 0.00 | -0.00 | -0.02 | 0.01 | 0.01 | 0.04 | -0.00 | -0.01 | 0.00 | -0.00 | -0.02 |
| 0.00 | 0.01 | 0.00 | 0.02 | 0.03 | 0.06 | 0.04 | 0.00 | 0.00 | 0.00 | 0.01 | 0.01 |
| -0.00 | 0.00 | 0.00 | 0.01 | 0.02 | 0.04 | 0.01 | 0.00 | 0.00 | -0.00 | 0.01 | 0.00 |
| 0.01 | 0.00 | 0.00 | 0.01 | 0.06 | 0.14 | 0.03 | 0.00 | 0.00 | -0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.01 | 0.02 | 0.06 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

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](#_bookmark61) demonstrates some ex- amples 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 grammati- cal 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 ex- tract 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 un- grammatical 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 un- grammatical sentence and obtain the average drop over all minimal edited pairs.

**Results and Discussion** Results are visualized in Fig [4](#_bookmark8). In general, We ﬁnd that the decrease of like- lihood 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

√

0.9

Corrupted

l

Ori gi na

**these group** of people based on their income ceil- ings .

*x*

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 out- weighs the cost of research and development .

The inexpensive fuel cost and the sheer volume of

0.8

0.7

acc

0.6

0.5

0.0 0.2 0.4 0.6 0.8 1.0

proporti on

energy produced by **the** nuclear **reactor** far out- weighs the cost of research and development . produced by a (the) nuclear reactor

*x*

0.05 -0.02 - 0.31 0.42

Table 5: Examples with ArtOrDet. We show the min- imal edit pairs and the likelihood decrease of each to- ken within two tokens away from the errors. Wrong de- terminers 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.](#_bookmark27), [2019](#_bookmark27)), we suspect that the presence of an error token will mostly affect its neighbor- ing tokens (both in terms of syntactic and physical neighbors). This is consistent with our observation in Fig [4](#_bookmark8) 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](#_bookmark9) to further illustrate this point.

# Adversarial Training

Finally, we explore a data augmentation method based on the proposed grammatical error simula- tions. We apply the greedy search algorithm to introduce grammatical errors to the training exam- ples of a target task and retrain the model on the combination of original examples and the gener- ated examples. We take the MRPC ([Dolan and](#_bookmark25) [Brockett](#_bookmark25), [2005](#_bookmark25)) dataset as an example to demon- strate the results. We augment the training set of

Figure 5: Results of a data augmentation defense. The proportions indicate the amount of adversarial exam- ples 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.

MRPC with different proportions of adversarial ex- amples, ﬁne-tune BERT on the augmented training set and then evaluate on both the original develop- ment set and the corrupted development set.

Results are shown in Figure [5](#_bookmark10). 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 origi- nal validation accuracy. This fact also demonstrates that our simulated examples are potentially helpful for reducing the effect of grammatical errors.

# 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 facili- tating our evaluations. We studied three pre-trained language encoders, ELMo, BERT, and RoBERTa and concentrated on three aspects of their abili- ties 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 er- rors. This study shed light on understanding the behaviors of language encoders against grammati- cal 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 anal-](https://openreview.net/forum?id=BJh6Ztuxl) [ysis of sentence embeddings using auxiliary pre-](https://openreview.net/forum?id=BJh6Ztuxl) [diction tasks](https://openreview.net/forum?id=BJh6Ztuxl). In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Pro- ceedings*.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. 2018. [Generating natural language adver-](https://www.aclweb.org/anthology/D18-1316/) [sarial examples](https://www.aclweb.org/anthology/D18-1316/). In *Proceedings of the 2018 Con- ference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - Novem- ber 4, 2018*, pages 2890–2896.

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

Timothy Baldwin, Trevor Cohn, and Yitong Li. 2017. [Robust training under linguistic adversity](https://www.aclweb.org/anthology/E17-2004/). In *Pro- ceedings of the 15th Conference of the European Chapter of the Association for Computational Lin- guistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 21–27.

Yonatan Belinkov and Yonatan Bisk. 2018. [Synthetic](https://openreview.net/forum?id=BJ8vJebC-) [and natural noise both break neural machine trans-](https://openreview.net/forum?id=BJ8vJebC-) [lation](https://openreview.net/forum?id=BJ8vJebC-). In *6th International Conference on Learn- ing Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Steven Bird and Edward Loper. 2004. [NLTK: the natu-](https://www.aclweb.org/anthology/P04-3031/) [ral language toolkit](https://www.aclweb.org/anthology/P04-3031/). In *Proceedings of the 42nd An- nual Meeting of the Association for Computational Linguistics, Barcelona, Spain, July 21-26, 2004 - Poster and Demonstration*.

Jill Burstein, Christy Doran, and Thamar Solorio, ed- itors. 2019. [*Proceedings of the 2019 Conference*](https://www.aclweb.org/anthology/volumes/N19-1/)[*of the North American Chapter of the Association*](https://www.aclweb.org/anthology/volumes/N19-1/)[*for Computational Linguistics: Human Language*](https://www.aclweb.org/anthology/volumes/N19-1/)[*Technologies, NAACL-HLT 2019, Minneapolis, MN,*](https://www.aclweb.org/anthology/volumes/N19-1/)[*USA, June 2-7, 2019, Volume 1 (Long and Short Pa-*](https://www.aclweb.org/anthology/volumes/N19-1/)[*pers)*](https://www.aclweb.org/anthology/volumes/N19-1/). Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robin- son. 2014. [One billion word benchmark for measur-](http://www.isca-speech.org/archive/interspeech_2014/i14_2635.html) [ing progress in statistical language modeling](http://www.isca-speech.org/archive/interspeech_2014/i14_2635.html). In *IN- TERSPEECH 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 *Black- BoxNLP@ACL*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Lo¨ıc Barrault, and Antoine Bordes. 2017. [Supervised](https://www.aclweb.org/anthology/D17-1070/)

[learning of universal sentence representations from](https://www.aclweb.org/anthology/D17-1070/) [natural language inference data](https://www.aclweb.org/anthology/D17-1070/). In *Proceedings of the 2017 Conference on Empirical Methods in Nat- ural Language Processing, EMNLP 2017, Copen- hagen, Denmark, September 9-11, 2017*, pages 670–

680.

Alexis Conneau, Germa´n Kruszewski, Guillaume Lam- ple, Lo¨ıc Barrault, and Marco Baroni. 2018. [What](https://doi.org/10.18653/v1/P18-1198) [you can cram into a single $&!#\* vector: Probing](https://doi.org/10.18653/v1/P18-1198) [sentence embeddings for linguistic properties](https://doi.org/10.18653/v1/P18-1198). In *Proceedings of the 56th Annual Meeting of the As- sociation for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1:*

[*\*](https://doi.org/10.18653/v1/P18-1198)

*Long Papers*, pages 2126–2136.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. [Building a large annotated corpus of learner](https://www.aclweb.org/anthology/W13-1703/) [english: The NUS corpus of learner english](https://www.aclweb.org/anthology/W13-1703/). 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 Car- bonell, Quoc Le, and Ruslan Salakhutdinov. 2019. [Transformer-XL: Attentive language models beyond](https://doi.org/10.18653/v1/P19-1285) [a ﬁxed-length context](https://doi.org/10.18653/v1/P19-1285). In *Proceedings of the 57th Annual Meeting of the Association for Computa- tional 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](https://www.aclweb.org/anthology/N19-1423/) [deep bidirectional transformers for language under-](https://www.aclweb.org/anthology/N19-1423/) [standing](https://www.aclweb.org/anthology/N19-1423/). In ([Burstein et al.](#_bookmark17), [2019](#_bookmark17)), pages 4171– 4186.

William B. Dolan and Chris Brockett. 2005. [Automati-](https://www.aclweb.org/anthology/I05-5002/) [cally constructing a corpus of sentential paraphrases](https://www.aclweb.org/anthology/I05-5002/). 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 abili-](http://arxiv.org/abs/1901.05287) [ties](http://arxiv.org/abs/1901.05287). *CoRR*, abs/1901.05287.

Ganesh Jawahar, Benoˆıt Sagot, and Djame´ Seddah. 2019. [What does BERT learn about the structure of](https://www.aclweb.org/anthology/P19-1356/) [language?](https://www.aclweb.org/anthology/P19-1356/) In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Vol- ume 1: Long Papers*, pages 3651–3657.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. [Is BERT really robust? natural](http://arxiv.org/abs/1907.11932) [language attack on text classiﬁcation and entailment](http://arxiv.org/abs/1907.11932). *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. Bow- man, and Ellie Pavlick. 2019. [Probing what differ-](https://www.aclweb.org/anthology/S19-1026/) [ent NLP tasks teach machines about function word](https://www.aclweb.org/anthology/S19-1026/) [comprehension](https://www.aclweb.org/anthology/S19-1026/). In *Proceedings of the Eighth Joint*

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

Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A](http://arxiv.org/abs/1412.6980) [method for stochastic optimization](http://arxiv.org/abs/1412.6980). In *3rd Inter- national 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-](https://transacl.org/ojs/index.php/tacl/article/view/972) [sensitive dependencies](https://transacl.org/ojs/index.php/tacl/article/view/972). *TACL*, 4:521–535.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Man- dar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. [Roberta: A robustly optimized BERT pretraining ap-](http://arxiv.org/abs/1907.11692) [proach](http://arxiv.org/abs/1907.11692). *CoRR*, abs/1907.11692.

Alison Lui, Antonios Anastasopoulos, and David Chi- ang. 2018. [Neural machine translation of text from](http://arxiv.org/abs/1808.06267) [non-native speakers](http://arxiv.org/abs/1808.06267). *CoRR*, abs/1808.06267.

Rebecca Marvin and Tal Linzen. 2018. [Targeted syn-](https://www.aclweb.org/anthology/D18-1151/) [tactic evaluation of language models](https://www.aclweb.org/anthology/D18-1151/). In *Proceed- ings 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 rep-](https://www.aclweb.org/anthology/N18-1202/) [resentations](https://www.aclweb.org/anthology/N18-1202/). In *Proceedings of the 2018 Confer- ence of the North American Chapter of the Associ- ation for Computational Linguistics: Human Lan- guage Technologies, NAACL-HLT 2018, New Or- leans, 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 pre-](https://www.aclweb.org/anthology/W19-4302/) [trained representations to diverse tasks](https://www.aclweb.org/anthology/W19-4302/). In *Proceed- ings of the 4th Workshop on Representation Learn- ing 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 inter-](https://www.aclweb.org/anthology/P06-1055/) [pretable tree annotation](https://www.aclweb.org/anthology/P06-1055/). In *ACL 2006, 21st Interna- tional Conference on Computational Linguistics and 44th Annual Meeting of the Association for Compu- tational 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](https://www.aclweb.org/anthology/D16-1264/) [machine comprehension of text](https://www.aclweb.org/anthology/D16-1264/). In *Proceedings of*

*the 2016 Conference on Empirical Methods in Nat- ural 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](https://doi.org/10.18653/v1/P18-1079) [debugging nlp models](https://doi.org/10.18653/v1/P18-1079). pages 856–865.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. [Introduction to the conll-2003 shared task:](http://arxiv.org/abs/cs.CL/0306050) [Language-independent named entity recognition](http://arxiv.org/abs/cs.CL/0306050). *CoRR*, cs.CL/0306050.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. [Recursive deep mod-](https://www.aclweb.org/anthology/D13-1170/) [els for semantic compositionality over a sentiment](https://www.aclweb.org/anthology/D13-1170/) [treebank](https://www.aclweb.org/anthology/D13-1170/). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Process- ing, EMNLP 2013, 18-21 October 2013, Grand Hy- att 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](https://www.aclweb.org/anthology/P19-1452/). In *Proceedings of the 57th Conference of the Associ- ation for Computational Linguistics, ACL 2019, Flo- rence, 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](https://openreview.net/forum?id=SJzSgnRcKX) [context? probing for sentence structure in contextu-](https://openreview.net/forum?id=SJzSgnRcKX) [alized word representations](https://openreview.net/forum?id=SJzSgnRcKX). In *International Con- ference 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](http://papers.nips.cc/paper/7181-attention-is-all-you-need) [you need](http://papers.nips.cc/paper/7181-attention-is-all-you-need). In *Advances in Neural Information Pro- cessing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 Decem-*

*ber 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 plat-](https://openreview.net/forum?id=rJ4km2R5t7) [form for natural language understanding](https://openreview.net/forum?id=rJ4km2R5t7). In *7th International Conference on Learning Representa- tions, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Alex Wang, Ian F. Tenney, Yada Pruksachatkun, Katherin Yu, Jan Hula, Patrick Xia, Raghu Pappa- gari, Shuning Jin, R. Thomas McCoy, Roma Pa- tel, Yinghui Huang, Jason Phang, Edouard Grave, Haokun Liu, Najoung Kim, Phu Mon Htut, Thibault F’evry, Berlin Chen, Nikita Nangia, Anhad Mo- hananey, 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 en-](http://arxiv.org/abs/1901.03438) [coders with acceptability judgments](http://arxiv.org/abs/1901.03438). *CoRR*, abs/1901.03438.

Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Ha- gen 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. [Investi-](http://arxiv.org/abs/1909.02597) [gating bert’s knowledge of language: Five analysis](http://arxiv.org/abs/1909.02597) [methods with npis](http://arxiv.org/abs/1909.02597). *CoRR*, abs/1909.02597.

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

Adina Williams, Nikita Nangia, and Samuel R. Bow- man. 2018. [A broad-coverage challenge corpus](https://www.aclweb.org/anthology/N18-1101/) [for sentence understanding through inference](https://www.aclweb.org/anthology/N18-1101/). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computa- tional 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, Pier- ric Cistac, Tim Rault, R’emi Louf, Morgan Funtow- icz, and Jamie Brew. 2019. Huggingface’s trans- formers: State-of-the-art natural language process- ing. *ArXiv*, abs/1910.03771.

# Downstream Task Details

We test on four language understanding and a se- quence labeling datasets. Statistics of these datasets are listed in Table [6](#_bookmark57).

**MRPC** The Microsoft Research Paraphrase Cor- pus (MRPC) ([Dolan and Brockett](#_bookmark25), [2005](#_bookmark25)) is a para- phrase detection task which aims to predict a binary label for whether two sentences are semantically equivalent.

**MNLI** The Multi-Genre Natural Language In- ference Corpus (MNLI) ([Williams et al.](#_bookmark52), [2018](#_bookmark52)) 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 mis- matched (cross-domain) sections.

**QNLI** The Question-answering NLI task (QNLI) is recasted from the Stanford Question Answering Dataset ([Rajpurkar et al.](#_bookmark39), [2016](#_bookmark39)), 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.](#_bookmark42), [2013](#_bookmark42))) 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](#_bookmark41) [Meulder](#_bookmark41), [2003](#_bookmark41)) is a token level sequence labeling task to recognize four types of named entity: per- sons, locations, organizations and names of miscel- laneous entities that do not belong to the previous three groups.

# Model Details

## 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 trans- former 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.](#_bookmark53) ([2019](#_bookmark53)) 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 BiL- STM with cross-sentence attention mechanism as implemented in *jiant*. ([Wang et al.](#_bookmark48), [2019b](#_bookmark48)).

## 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](#_bookmark30), [2015](#_bookmark30)) 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](#_bookmark30), [2015](#_bookmark30)) 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.

## 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* = *{hi , hi , ..., hi }* and outputs a sen-

1

2

*n*

tence representation *si*:

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

*si* = Σ*n*

*j*=1

*αjhi*

*j*

(1)

*αj* = *softmax*(*vT tanh*(*Wahi* )) (2)

*b j*

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 to- ken) of the target sentence being attacked.

where *Wa* is a weight matrix and *vb* is a vector of parameters. *si* 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](#_bookmark30), [2015](#_bookmark30)) 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.

# Attack Algorithms

We conduct three searching algorithms, *greedy search*, *beam search*, *genetic algorithm* in adver- sarial attacks based on the real errors on NUCLE ([Dahlmeier et al.](#_bookmark22), [2013](#_bookmark22)). 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](#_bookmark59), [2](#_bookmark63) and [3](#_bookmark64) are detailed descriptions of greedy attack, beam search attack, and genetic algorithm attack, respectively.

**Algorithm 1** Greedy attack 10cm

*{ }*

**Input:** Original sentence *Xori* = *w*1*, w*2*, ..., wn* , ground

truth prediction *Yori*, target model *F* , all confusion sets *P* , budget *b*.

**Output:** Adversarial example *Xadv*. 1: Initialization: *Xadv Xori*

*←*

2: **for** each *wi* in *Xori* **do**

3: Delete *wi* and compute the drop of likelihood on *Yori* to obtain the importance score of *wi*, denoted as *Sw』* .

4: Apply all substitutions of *P* to *wi*. Obtain the

operation pool of *wi*, denoted as *Wsub*.

*i*

5: **end for**

6:

7: Get the index list of *Xori* according to the decrease order of token importance: *I argsortw』 XóT』* (*Sw』* )

8: *← e*

**for** each *i* in *I* **do**

9: *pori ← F* (*Xadv* )*|Y* =*YóT』*

10: **for** each *w′* in *Wsub* **do**

*i*

11: Substitute *wi* with *w′* in *Xadv* (or swap their

# Probing Model Ability in Identifying Errors

12:

13:

positions),

*Yadv argmaxF* (*Xadv*), *padv F* (*Xadv* ) *Y* =*YóT』*

*← |*

*←*

**if** not *Yori* = *Yadv* **then return** *Xadv*

## The Sentence-level Binary Classiﬁcation Task

Table [7](#_bookmark65) shows complete results for probing individ- ual layers of ELMo, BERT, and RoBERTa across eight error types in the sentence-level binary classi- ﬁcation task. We ﬁx the parameters of pre-trained encoders and train a self-attention classiﬁer for each layer to judge the binary linguistic accept- ability 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.

## The Token-level Error Locating Task

Table [8](#_bookmark66) shows complete results for probing individ- ual 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 atten- tion weights of self-attention layers and see if error tokens are included.

# Case Study of Locating Error Positions

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

14: **else**

15: **if** *padv < pori* **then**

16: *wselect w′* , *pori padv*

*← ←*

17: **end if**

18: **end if**

19: **end for**

20: **if** the number of iterations exceed *b* **then return** *Xori*

21: **end if**

22: Substitute *wi* with *wselect* in *Xadv*, 23: **end for**

24: **return** *Xori*

# 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. Mid- dle layers of BERT perform the best. We further observe that some attention heads might be asso- ciated 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 *Xori* = *w*1*, w*2*, ..., wn* , ground truth prediction *Yori*, target model *F* , all confusion sets *P* , budget *b*, beam size *bm*.

*{ }*

**Output:** Adversarial example *Xadv*.

1: Initialization: *bestBeam* copy *Xori* for *bm* times. 2: **for** each *wi* in *Xori* **do**

*←*

3: Delete *wi* and compute the drop of likelihood on *Yori* to obtain the importance score of *wi*, denoted as *Sw』* .

4: Apply all substitutions of *P* to *wi*. Obtain the

operation pool of *wi*, denoted as *Wsub*.

*i*

5: **end for**

6:

7: Get the index list of *Xori* according to the decrease order of token importance: *I ← argsortw』eXóT』* (*Sw』* )

8: **for** each *w′* in *Wsub* **do**

*I*[0]

9: Substitute *wi* with *w′* in *Xori* (or swap their

**Algorithm 3** Genetic attack

**Input:** Original sentence *Xori* = *w*1*, w*2*, ..., wn* , ground truth prediction *Yori*, target model *F* , all confusion sets *P* , budget *b*, population size *ps*, generation size *G*.

*{ }*

**Output:** Adversarial example *Xadv*.

1: Initialize the ﬁrst generation with empty set: *P* 0 . 2: **for** each *wi* in *Xori* **do**

*← 份*

3: Apply all substitutions of *P* to *wi*. Obtain the operation pool of *wi*, denoted as *Wsub*.

*i*

4: **end for**

5: **for** *i* = 1*,* 2*,* 3*, ..., ps* **do**

6: Randomly select a position *j* and an operation from

*Wsub* to modify *Xori*. Then add to *P* 0.

*j*

7: **end for**

8:

9: **for** *g* = 1*,* 2*,* 3*, ..., G* 1 **do**

*-*

10: **for** *i* = 1*,* 2*,* 3*, ..., ps* **do**

11: *Yadv ← argmaxF* (*P* ),

*g\_*1

positions)

10: *Yadv ← argmaxF* (*Xori*),

11: *← |*

*i*

*padv F* (*Pg\_*1) *Y* =*Y*

*← |*

*i óT』*

12: **if** not *Yadv* = *Yori* **then return** *Pg\_*1

*padv F* (*Xori*) *Y* =*YóT』*

**if** not *Yori* = *Yadv* **then return** *Xori*

12: **else**

13: **else**

14: *Xelite ← argmin*(*p*

*i*

*adv* )

15: *Pg ← {Xelite}*

13: *topBeam* Record top-*bm* examples with the lowest *padv*

*←*

14: **end if**

16:

1

*prob ←* Normalize sample probability with

*F* (*Pg\_*1)

*i*

15: **end for**

16:

17: **for** *i* = 2*,* 3*, ..., ps* **do**

18: Sample parent1 from *Pg\_*1 with probs

17: *bestBeam topBeam*

*←*

18: **for** each *i* in *I/I*[0] **do**

19: *pori ← F* (*Xadv*)*|Y* =*YóT』*

19:

20:

*prob*

Sample parent2 from *P prob*

*←*

*←*

*j*

*g\_*1

with probs

20: *oplist*

*← {}*

21: **for** each *Xbeam* in *bestBeam* **do**

22: **for** each *w′* in *Wsub* **do**

*i*

21:

*child* Crossover(parent1, parent2) *childmut* Randomly select a position and an operation from *Wsub* to modify

23: Substitute *wi* with *w′* in *Xbeam* positions)

24: *Yadv ← argmaxF* (*Xbeam*),

*padv ← F* (*Xbeam*)*|Y* =*Y*

(or swap their

*child*

22: *Pg childmut*

*i*

*←*

23: **end for**

24: **end if**

25: **if** not *Y*

26: **else**

*ori*

*óT』*

= *Yadv* **then return**

*Xbeam*

25: **end for**

26: **end for**

27: Add *op* (*w′ , padv, Xbeam*) to *oplist* 28: **end if**

*←*

29: **end for**

30: **end for**

31: **if** number of iterations exceed *b* **then return** *Xori*

32: **end if**

33: Select the top-*bm op*s in *oplist* with lowest *op.padv*.

Update *bestBeam* with each *op.Xbeam*.

34: **end for**

35: **return** *Xori*

27: **return** *Xori*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **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.

layer1 layer2 layer3 layer4 layer5 layer6 layer7 layer8 layer9 layer10 layer11 layer12

[CLS]

11

attacks

,

died

for

the

plane

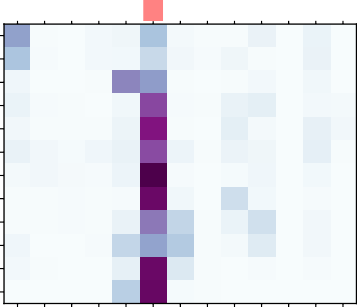
crash

near

Buffalo

.

[SEP]

Prep Error

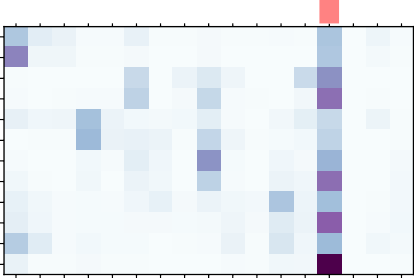
[CLS]

eating

digeste

[SEP]

layer1 layer2 layer3 layer4 layer5 layer6 layer7 layer8 layer9 layer10 layer11 layer12

ArtOrDet Error

[CLS]

That

might

help

reduce

the

risk

of

a

bank

running

out

of

the

capital

.

[SEP]

layer1 layer2 layer3 layer4 layer5 layer6 layer7 layer8 layer9 layer10 layer11 layer12

And

after

they

the

food

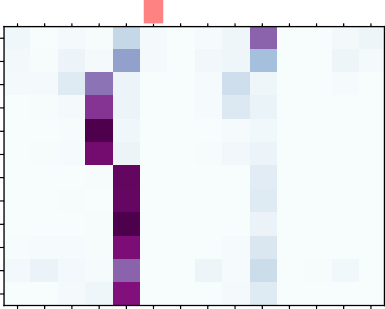
,

they

the

food

.

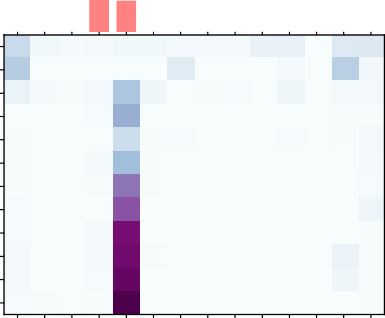
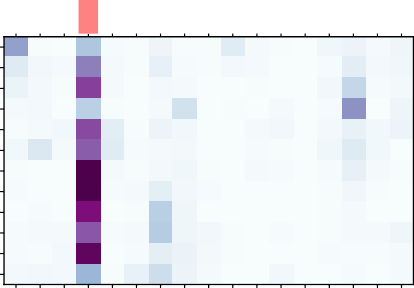
Vform Error

certificatio

consumers

sustainabl

layer1 layer2 layer3 layer4 layer5 layer6 layer7 layer8 layer9 layer10 layer11 layer12

Nn Error

[CLS]

[CLS]

His

It

only

has

rivals

,

been

ruling

even

right-wing

argued

candidate

that

Rodrigo

Man

Avila

is

,

making

shortly

it

afterwards

worse

conceded

defeat

.

.

[SEP]

[SEP]

layer1 layer2 layer3 layer4 layer5 layer6 layer7 layer8 layer9 layer10 layer11 layer12

[CLS]

We

needs

a

new

form

of

to

help

to

choose

seafood

.

[SEP]

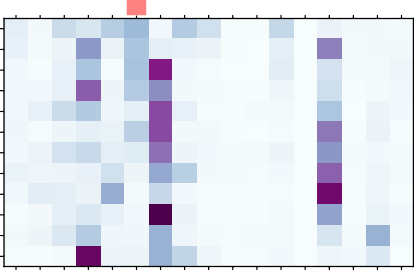
layer1 layer2 layer3 layer4 layer5

[CLS]

He

SVA Error

layer1 layer2 layer3 layer4 layer5 layer6 layer7 layer8 layer9 layer10 layer11 layer12

layer1 layer2 layer3 layer4 layer5

[CLS]

It

Worder Error

is

among

the

small

percenta

of

Pygmies

who

can

read

but

write

.

[SEP]

will

be

Rooney

's

beginning

meeting

with

the

world

's

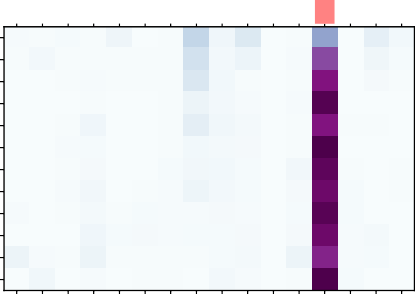
most

glamorous

team

.

[SEP]

layer6 layer6

layer7 layer8 layer9 layer10 layer11 layer12

Trans Error

layer7 layer8 layer9 layer10 layer11 layer12

Wchoice Error

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.

0.4

Acc.

0.2

prep

0.6

0.4

Acc.

0.2

sva

0.0

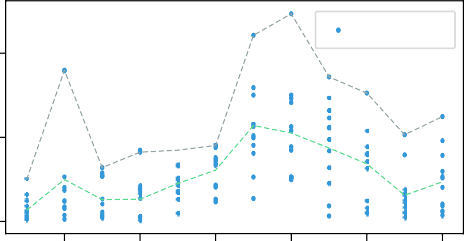
0.3

0.2

Acc.

0.1

2 4 6 8 10 12



10

10

Attn head

10

7

4

6

6

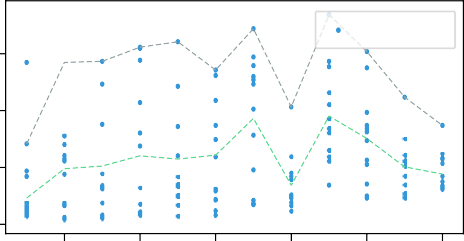
12 4

6

3

2

artordet



3

7

12 2

7

2

Attn head 7

3

1

6

2

3

0.0

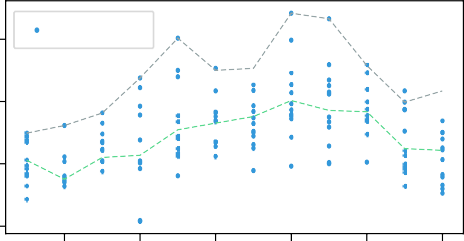
0.6

0.4

Acc.

0.2

2 4 6 8 10 12



2

6

Attn head 5

9

12 12

4

5

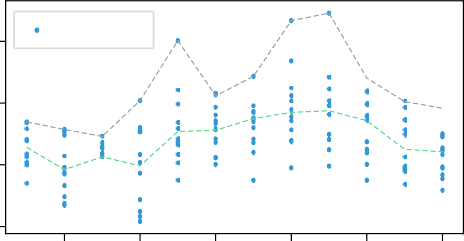
1

12

4

5

vform



2

6

Attn head

5

4

12

9

10

6

4

12

5

5

0.0

0.4

0.3

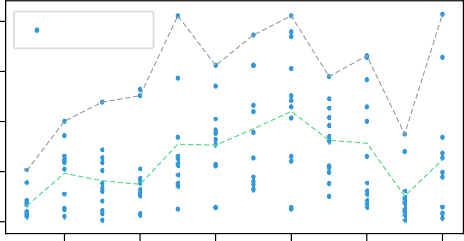
Acc.

0.2

0.1

2 4 6 8 10 12

trans



10

1

3

Attn head

3

6

11

1

5

4

2

6

5

0.0

0.6

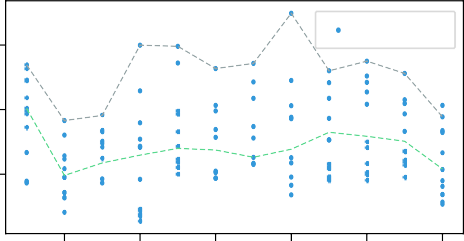
0.4

Acc.

0.2

2 4 6 8 10 12

nn



11

11 1

Attn head

2

1

6

6

8

11

3

11

6

0.0

0.6

0.4

Acc.

2 4 6 8 10 12

wchoice

0.6

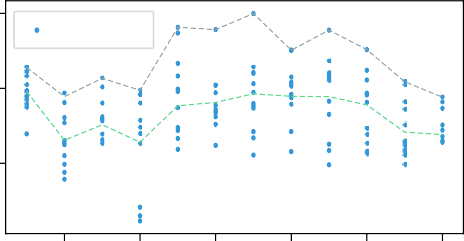
0.4

Acc.

2 4 6 8 10 12

worder

0.2



Attn head

1 10

5

6

2 10

7

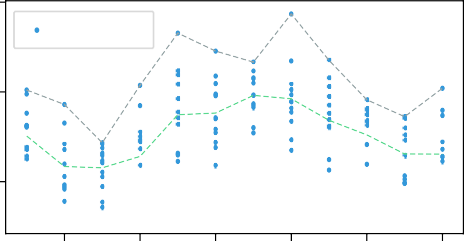
7

12

2

10

6



4

Attn head 7

8

4

9

12

9

11

2

10

10

1

0.2

2 4 6 8 10 12 2 4 6 8 10 12

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.