On the Robustness of Language Encoders against Grammatical Errors

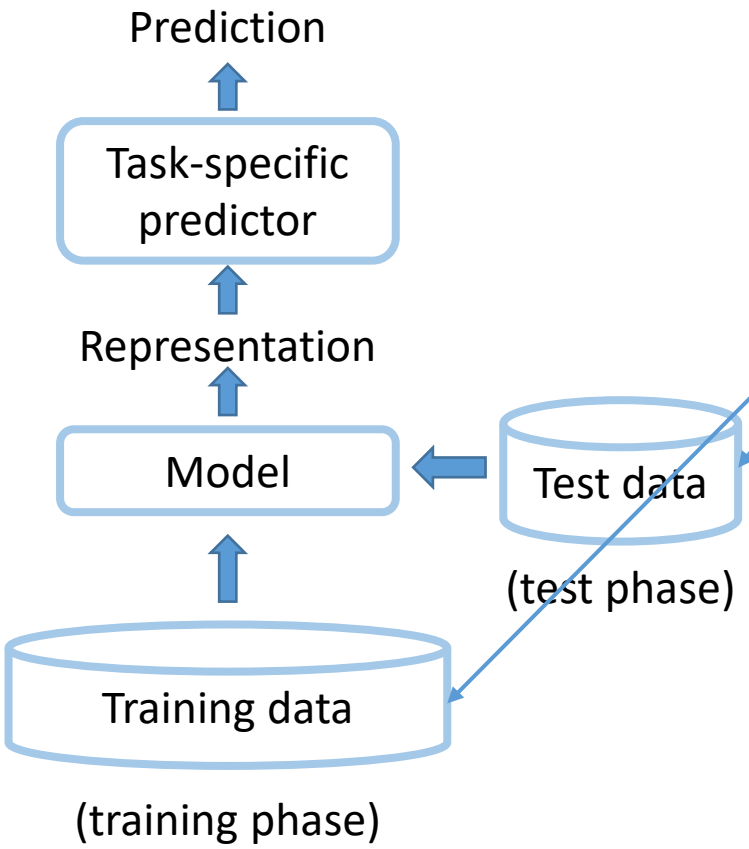
Fan Yin[1], Quanyu Long[2], Tao Meng[3], Kai-Wei Chang[3]

[1]Peking University [2]Shanghai Jiaotong University [3]UCLA

ACL 2020

Language Encoders for English Text

- Pre-trained encoder facilitates many NLP tasks
- Many variants: ELMo, BERT, RoBERTa…



A basic assumption: training and test data are in (almost) perfect English

How language encoders perform when confronted with grammatical errors?

Treating Grammatical Errors as Noise

- Frequently occur in materials of non-native speakers
- Resources: Grammatical Error Correction benchmarks

[Hwee Tou Ng et al. 2014]

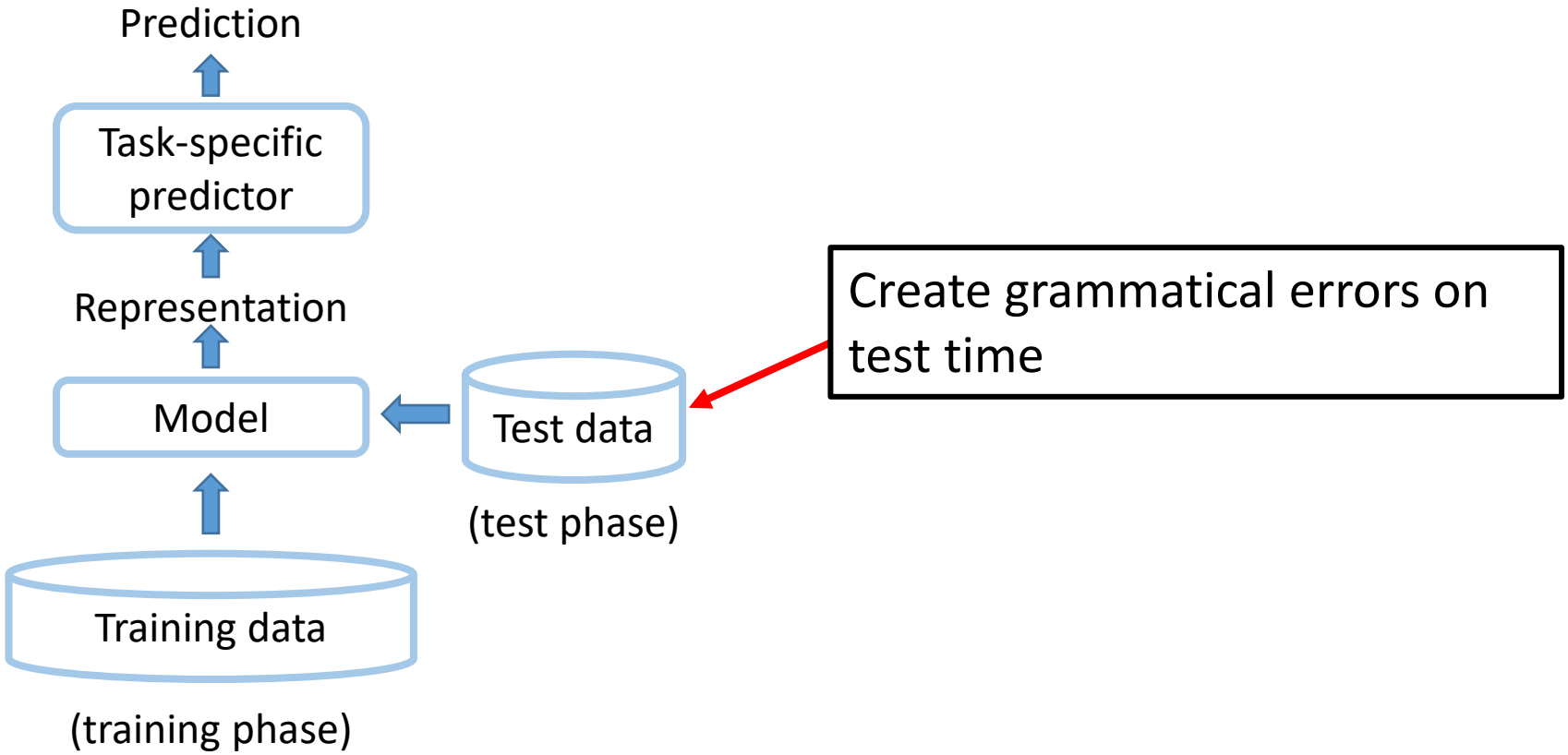
Prep	Preposition errors	This essay will [discuss about → discuss] whether a carrier should tell his relatives or not.

Error type

Ungrammatical sentence annotated with local edits

Key Contribution 1: Evaluate Language Encoders against Grammatical Errors

- Analyze how grammatical errors affect model behavior
- Understand if grammar structure is encoded



Key Contribution 2: Automatic Grammatical Error Simulation

Our automatic grammatical error simulator considers two scenarios: [1]

- Average case: conforms to the real error distribution estimated from an ESL corpus
- Worst case: analyzes the brittleness of models by treating grammatical errors as adversarial attacks

[1] Prior work manually construct new datasets as test data [Marvin and Linzen, 2018; Warstadt et al. 2019]

Outline

- Background & Motivation
- Grammatical Error Simulation
- Evaluation
- Summary

Grammatical Error Simulation

1. Collect and mimic the real error distribution

- Collect errors from NUCLE a grammatical error correction benchmark [Dahlmeieretal et al. 2013]
- Construct a pool of possible candidates

2. Inject errors

- Token-level transformation
- Probabilistic and worst-case transformation

Collect and Mimic Error Distribution

- Select frequent error types (Similar as [Lui et al. 2019])

Error type	Error Description	Confusion Set
ArtOrDet	Article/determiner errors	{ a, an, the, ø}
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, ø}
Trans	Link words/phrase errors	{and, but, so, however, as, that, thus, also, because, 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}

Both

Semantics

Syntax

Collect and Mimic Error Distribution

- Construct confusion sets for error types from an ESL corpus (Similar as [Lui et al. 2019])

p(error|correct)	a	An	the	ø
a		0.01	0.27	0.73
an	0.2		0.25	0.55
the	0.12	0.02		0.86
ø	0.13	0.02	0.84	

(This table is modified from http://www.cs.cmu.edu/~aanastas/research/GECNMT.pdf)

Inject Errors -- Average Case Analysis

- Sample an error type x
- Syntactic parse tree to decide a plausible position
- Sample a substitution from confusion sets of x

Inject Errors – Worst Case Analysis

- For each position, check all confusion sets for possible substitutions, maintain an operation set
- Using three search algorithms to select operations from operation sets
  - greedy search
  - beam search
  - genetic algorithm

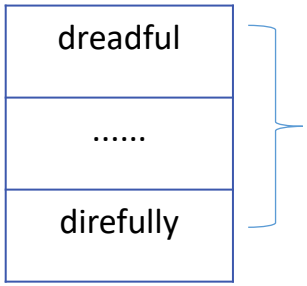
Inspired by the literature of adversarial attacks [Jin et al. 2020; Alzantot et al. 2018]

Example of Greedy Search

Input: it's of the quality of a lesser harrison ford movie - six days, seven nights, maybe, or that direful sabrina remake.
(from SST-2)

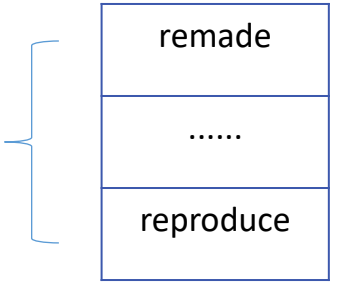
	that	direful	sabrina	remake
Step 1: rank token importance	4	1	10	2

Step 2:
try replacements in turn



The operation set of the word "direful"

The operation set of the word "remake"



Outline

- Background & Motivation
- Grammatical Error Simulation
- Evaluation
- Summary

Experiment Analysis

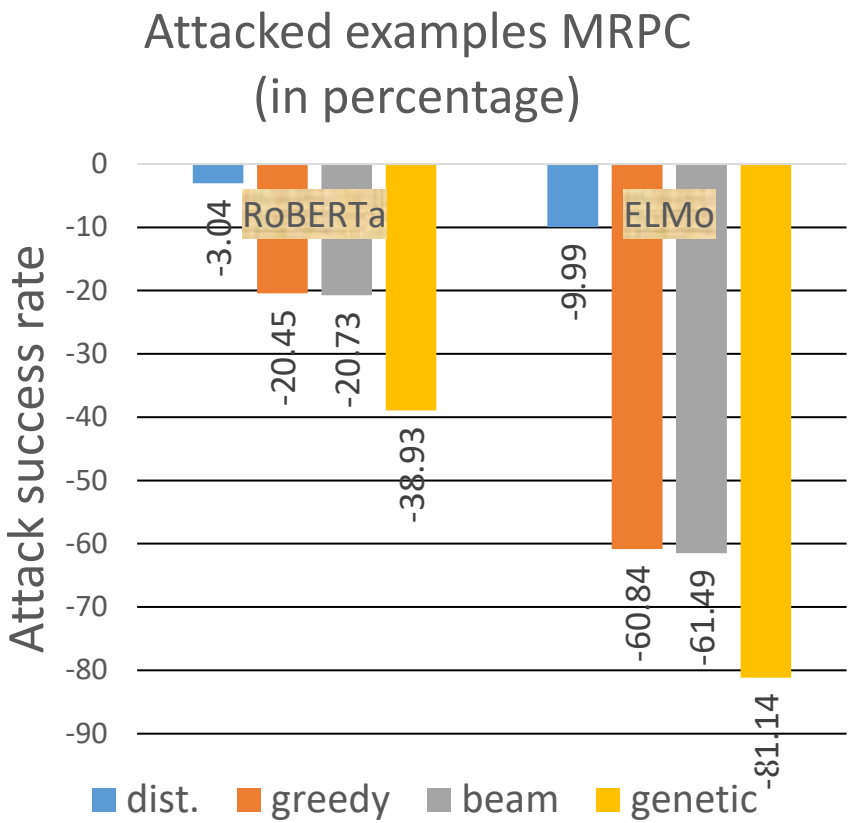
Our goal is to study

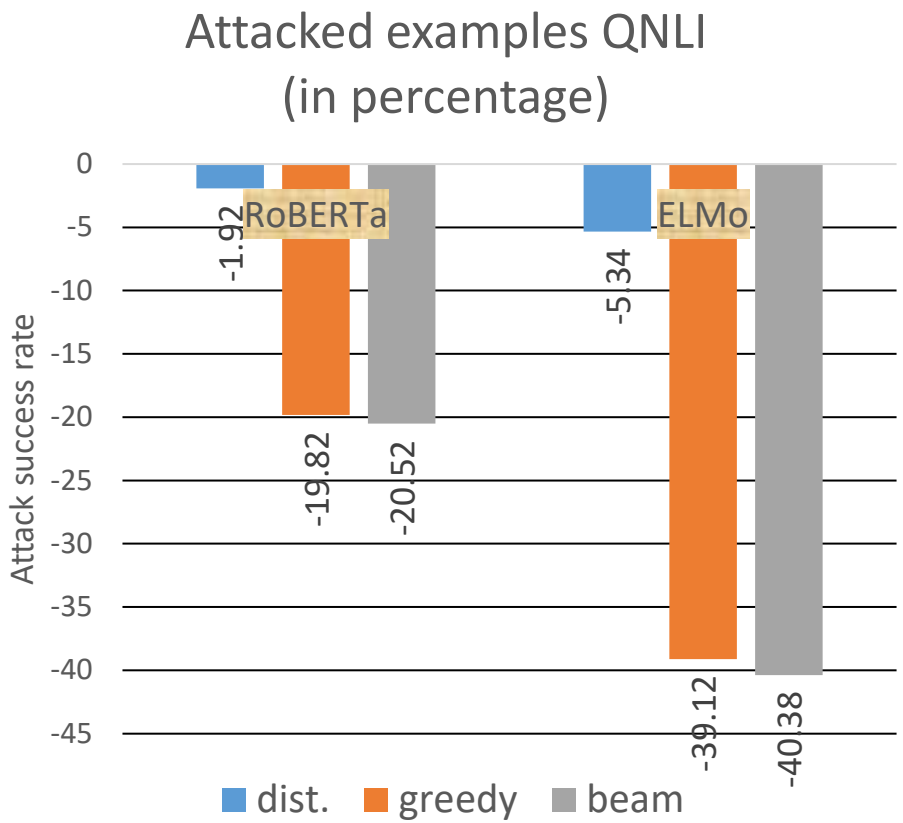
- How grammar errors affect performance on downstream tasks?
  - Are language encoders robust against perturbations?
  - Which error types affect the models the most?
  - Which downstream tasks are more sensitive?
- Investigate with probing tasks
  - How models capture grammatical errors with contexts?

Experiment Setup

- Language encoders: ELMo, **BERT, RoBERTa**, InferSent
- Downstream datasets: MRPC, MNLI,QNLI, SST-2, CONLL-2013 NER
- Probing tasks: **Masked LM**, binary linguistic acceptability, error location prediction

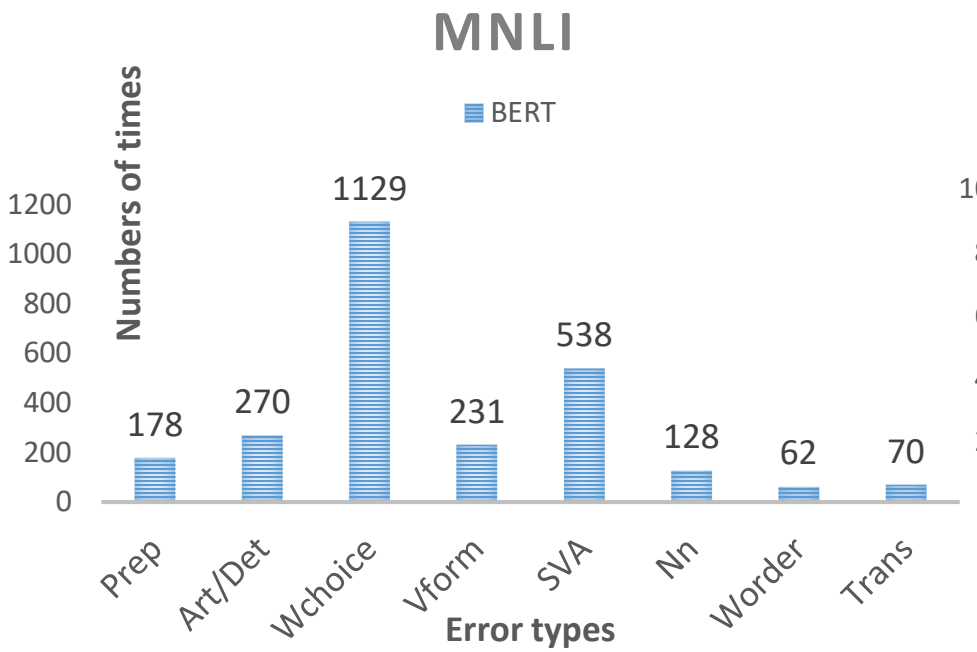
Downstream Task Evaluation

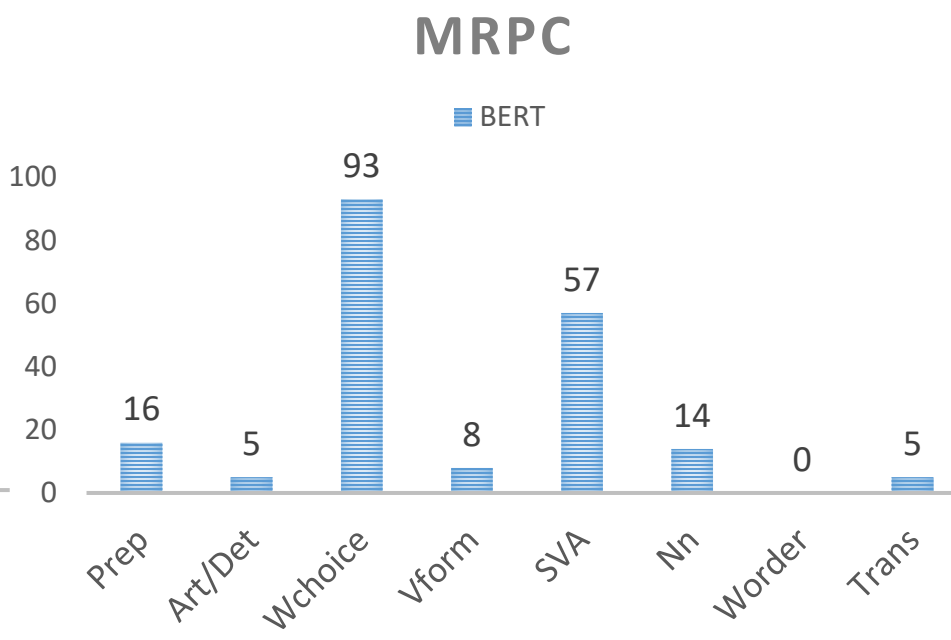




- The robustness of models varies
- RoBERTa is less sensitive to grammatical errors

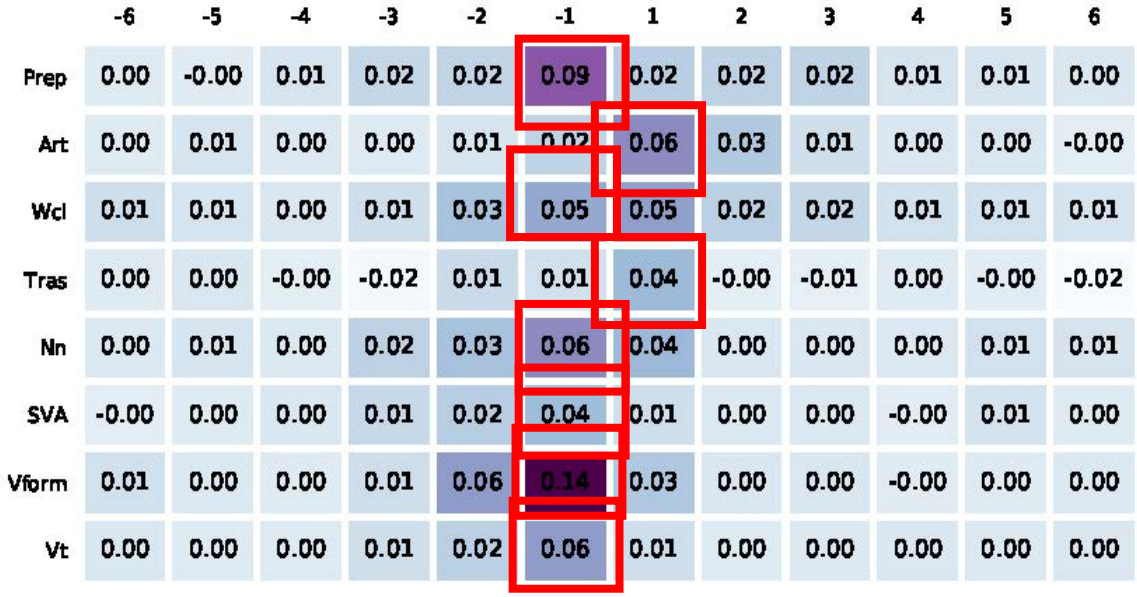
Error Types v.s. Model Performance





- Models are brittle to word choice (Wchoice) and subject-verb agreement errors (SVA)
- Relatively robust to word order errors (Worder)

Masked Language Model



	-6	-5	-4	-3	-2	-1	1	2	3	4	5	6
Prep	0.00	-0.00	0.01	0.02	0.02	0.09	0.02	0.02	0.02	0.01	0.01	0.00
Art	0.00	0.01	0.00	0.00	0.01	0.02	0.06	0.03	0.01	0.00	0.00	-0.00
Wcl	0.01	0.01	0.00	0.01	0.03	0.05	0.05	0.02	0.02	0.01	0.01	0.01
Tras	0.00	0.00	-0.00	-0.02	0.01	0.01	0.04	-0.00	-0.01	0.00	-0.00	-0.02
Nn	0.00	0.01	0.00	0.02	0.03	0.06	0.04	0.00	0.00	0.00	0.01	0.01
SVA	-0.00	0.00	0.00	0.01	0.02	0.04	0.01	0.00	0.00	-0.00	0.01	0.00
Vform	0.01	0.00	0.00	0.01	0.06	0.14	0.03	0.00	0.00	-0.00	0.00	0.00
Vt	0.00	0.00	0.00	0.01	0.02	0.06	0.01	0.00	0.00	0.00	0.00	0.00

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

Determiner-noun dependency

Summary

- We propose a new method to simulate grammatical errors, considering real errors and search algorithms in adversarial attacks
- We perform a systematical evaluation and analysis towards models based on our proposed method

Source code are available at:
https://github.com/uclanlp/ProbeGrammarRobustness
Thank you!