grammarly

Building sequence tagging approach to Grammatical Error Correction and Text Simplification

Oleksandr Skurzhanskyi, Kostia Omelianchuk, Jan 28, 2022

Outline of the talk

- Introduction
  - Few words about us
  - GEC task overview
- GEC task
- Sequence tagging approach
- GECToR for GEC
- Reusing GECToR on Text Simplification Task
- Q&A

Who we are?

Who we are?

Oleksandr Skurzhanskyi, Applied Research Scientist, Grammarly

Kostia Omelianchuk, Applied Research Scientist, Grammarly

What is Grammarly?

Grammarly's AI-powered writing assistant helps you make your communication clear and effective, wherever you type.

GEC task overview

GEC: Grammatical Error Correction

The goal of GEC task is to produce the grammatically correct sentence from the sentence with mistakes. Here we're talking about *English language* specifically.

Example:

Source: He go at school.

Target: He goes to school.

GEC: Data

TABLE 1
Statistics and properties of public GEC datasets.

Corpus	Component	# Sents	# Tokens	# Chars per sent	Sents Changed	# Ref	Error Type	Error Type	Proficiency	Topic	L1
NUCLE	-	57k	1.16M	115	38%	2	minimal	Labeled	Simplex	Simplex	Simplex
FCE	Train	28k	455k	74	62%	1	minimal	Labeled	Simplex	Diverse	Diverse
	Dev	2.1k	35k								
	Test	2.7k	42k								
Lang-8	-	1.04 M	11.86 M	56	42%	1-8	fluency	None	Diverse	Diverse	Diverse
JFLEG	Dev	754	14k	94	86	4	fluency	None	Diverse	Diverse	Diverse
	Test	747	13k								
W&I	Train	34.3k	628.7k	60	67%	1	-	Labeled	Diverse	Diverse	Diverse
	Dev	3.4k	63.9k	94	69%	1					
	Test	3.5k	62.5k	-	-	5					
LOCNESS	Dev	1k	23.1k	123	52%	1	-	Labeled	Diverse	Diverse	Simplex
	Test	1k	23.1k	-	-	5					

GEC: Shared task

- CoNLL-2014 (the test set is composed of 50 essays written by non-native English students; metric - M2Score)
- BEA-2019 (new data; the test set consists of 4477 sentences; metric - Errant)

GEC: Progress

6.2.1 Development of Approaches

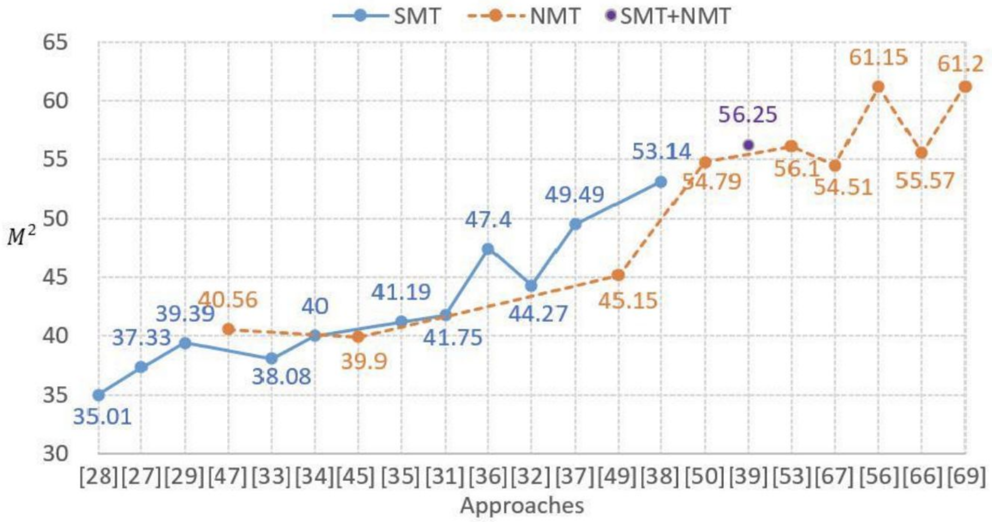
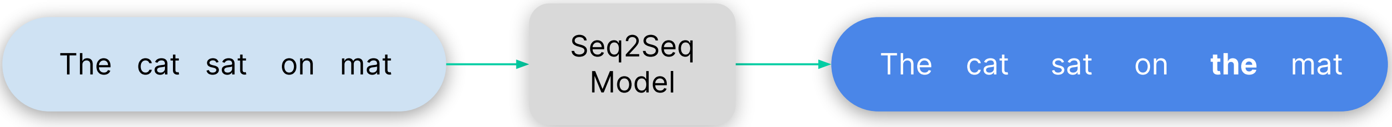


Fig. 4. Development of SMT based approaches and NMT based approaches.

Dominant approach in 2019

- GEC was treating as machine translation problem mostly
- Seq2Seq models are status quo for most sentence-level transduction tasks



- Drawbacks
  - Require large amounts of annotated parallel data - slow and expensive
  - Autoregressive decoding - not performant
  - Not controllable or explainable - unfit for real-world products

Sequence Tagging

This work was done in fall 2019

This part of the presentation is based on the paper written by:

- Kostiantyn Omelianchuk
- Oleksandr Skurzhanskyi
- Vitaliy Atrasevych
- Artem Chernodub

Sequence Tagging

We approach the GEC task as a sequence tagging problem. In this formulation for each token in the source sentence a GEC system should produce a tag (edit) which represent a required correction operation for this token.

For solving this problem we use non-autoregressive transformer-based model.

Basic transformations

KEEP keep the current token unchanged

DELETE delete current token

APPEND_t_i append new token t_i

REPLACE_t_i replace the current token with token t_i

g-transformations

CASE change the case of the current token

MERGE merge the current and the next token

SPLIT split the current token into two

NOUN NUMBER convert singular nouns to plurals and vice versa

VERB FORM change the form of regular/irregular verbs

Token-level transformations

A ten years old boy go school

A ⇨ A: $KEEP

ten ⇨ ten, -: $KEEP $MERGE_HYPH

years ⇨ year, -: $NOUN_NUMBER_SINGULAR $MERGE_HYPH

old ⇨ old: $KEEP

go ⇨ goes, to: $VERB_FORM_VB_VBZ $APPEND_{to}

school ⇨ school, .: $KEEP $APPEND_{.}

A ten-year-old boy goes to school.

Sequence Tagging: Pros

1. Generating tags is fully independent and easy to parallelize operation.
2. Smaller output vocabulary size compare to seq2seq models.
3. Less usage memory and faster inference compare to encoder-decoder models.

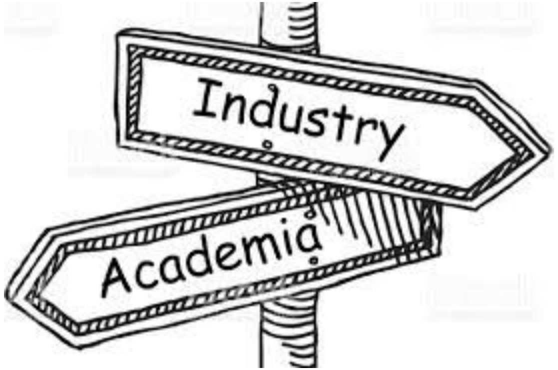
Sequence Tagging: Cons

1. Independent generating of tags relies on assumption that errors are independent between each other.
2. Word reordering is tricky for this architecture.

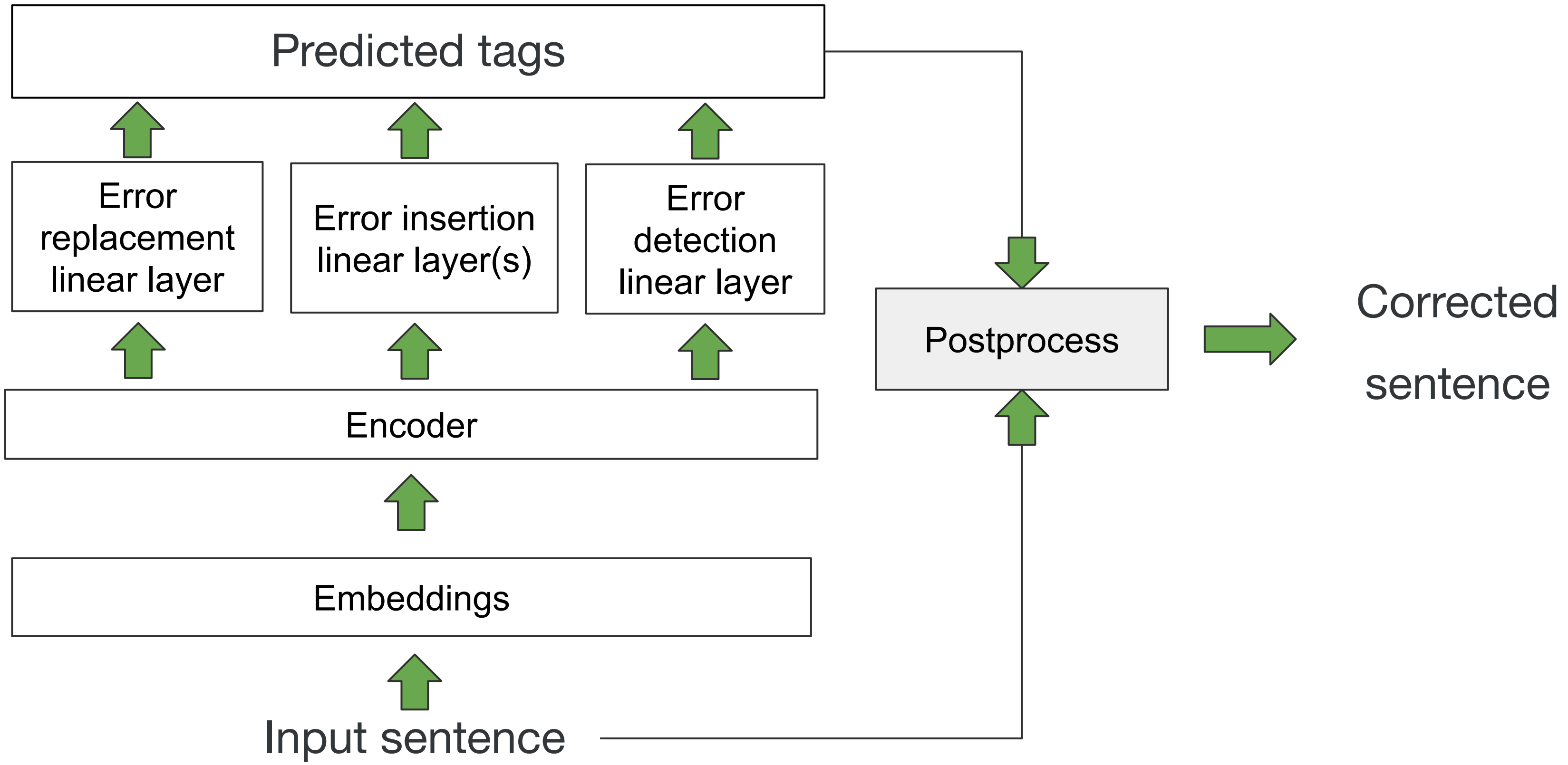
First baselines

Academia vs Industry

- different data (both training/evaluation)
- latency/throughput matters
- aggressive deadlines



Baseline architecture



Baseline hyperparameters

- Emedings (trainable random, glove, bert, distill-bert)
- Encoder (cnn, lstm, stacked-lstm, pass_through, transformers)
- Output layers (1-2 for insertions, 1 for replacement, 1 for detection)
- Output vocabulary size (100-50000)
- Other (dropouts, training schedule, tp/tn ratio, etc)

Baseline results

Baseline/m2 score	CoNLL-2014 (test)
Stacked LSTM (vocab_size=1000)	30.5
Stacked LSTM (vocab_size=1000; + g-transformations)	35.6
Stacked LSTM (vocab_size=1000; + g-transformations; + BERT embeddings)	46
Academic SOTA (single model) [2019]	61.3

Insights

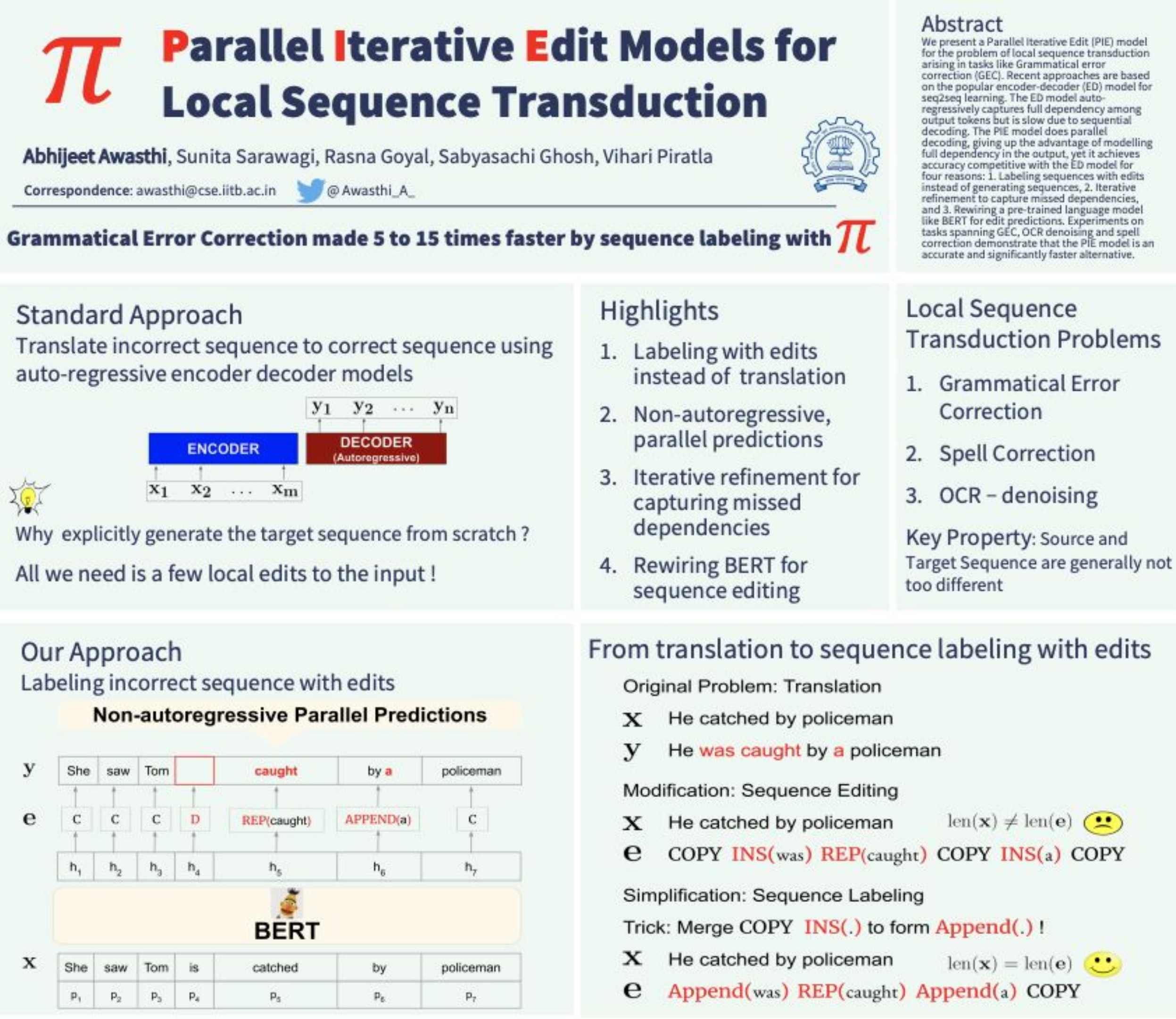
- Increasing size of output vocabulary did not help
- Adding BERT as emdebbings helped a lot
- Training BERT with small batch_size failed (didn’t converge);
  training with bigger batches required gradient accumulating

Similar approach

PIE paper

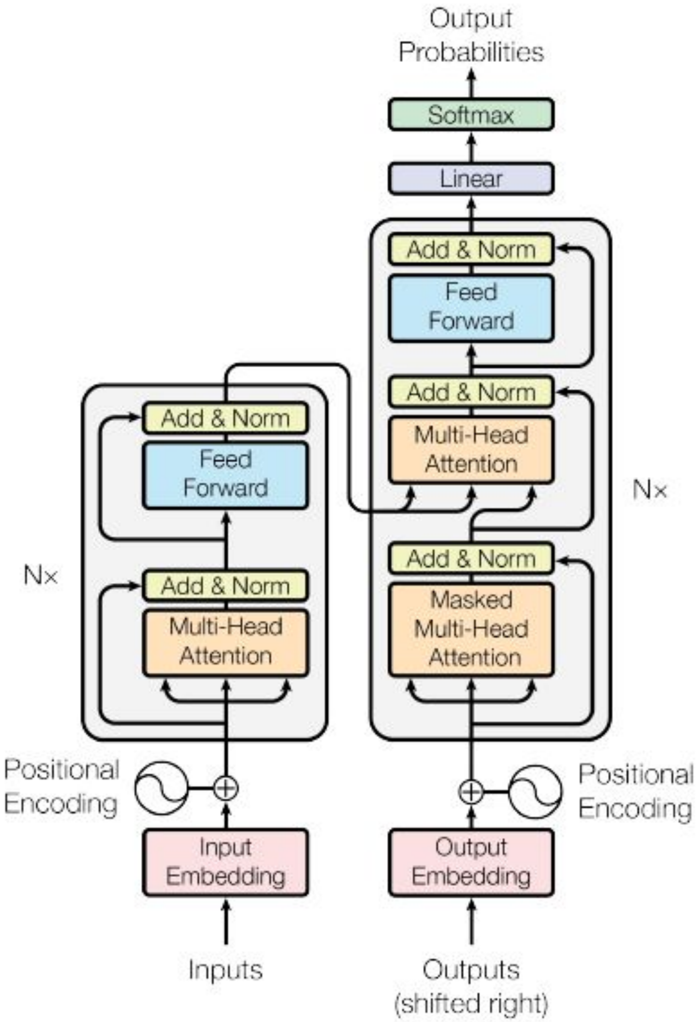
#	Methods	P	R	$F_{0.5}$
1	PIE	66.1	43.0	59.7
2	– Synthetic training	67.2	34.2	56.3
3	–Factorized-logits	66.4	32.8	55.1
4	–Append +Inserts	57.4	42.5	53.6
5	–Transformations	63.6	27.9	50.6
6	–LM Pre-init	48.8	18.3	36.6
7	PIE on BERT-Base	67.8	34.0	56.6

Our SOTA on NUCLE: 46



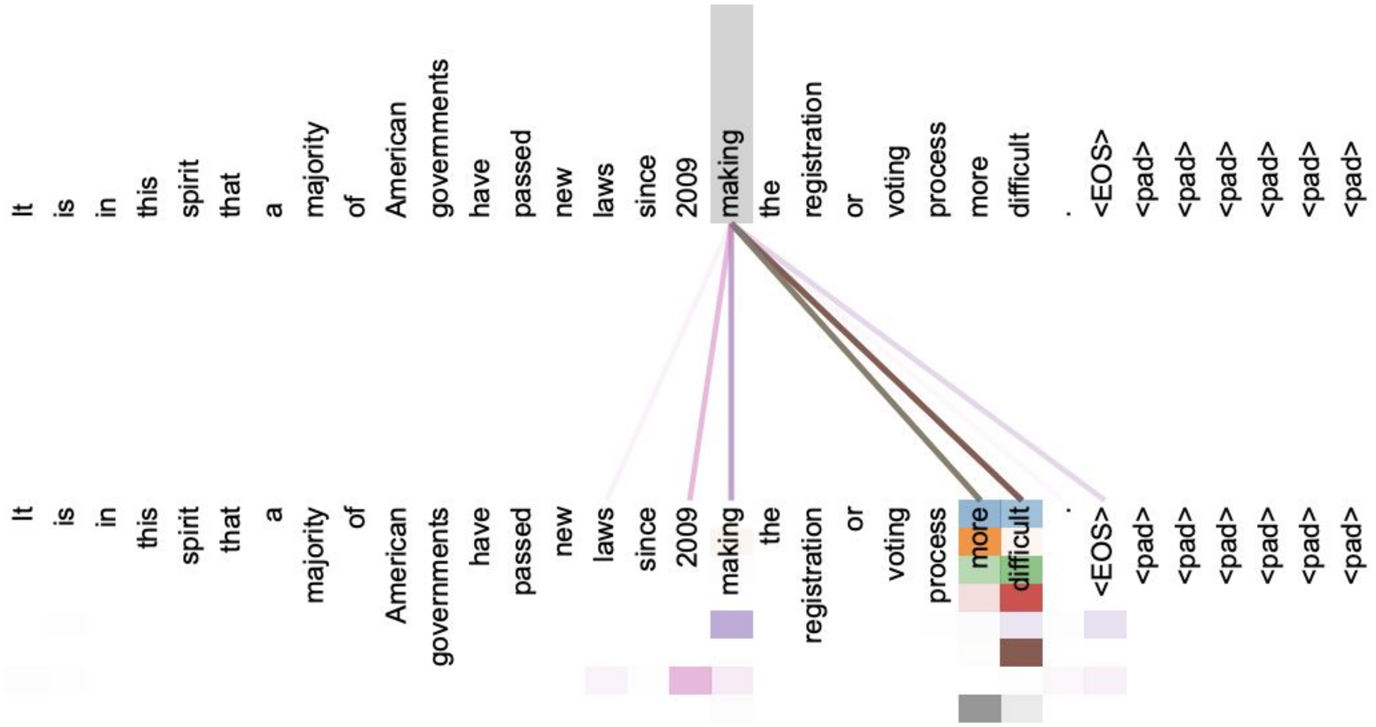
Transformers

Transformer

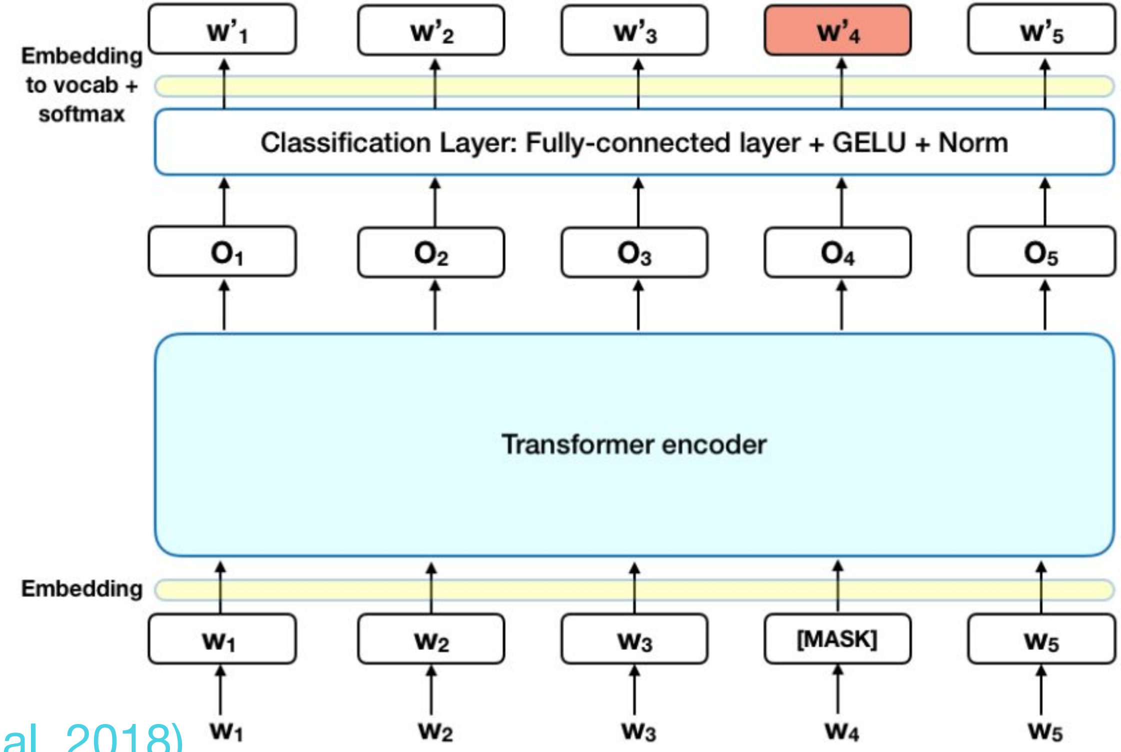


[Attention Is All You Need
(Vaswani et al. 2017)]

Multihead attention



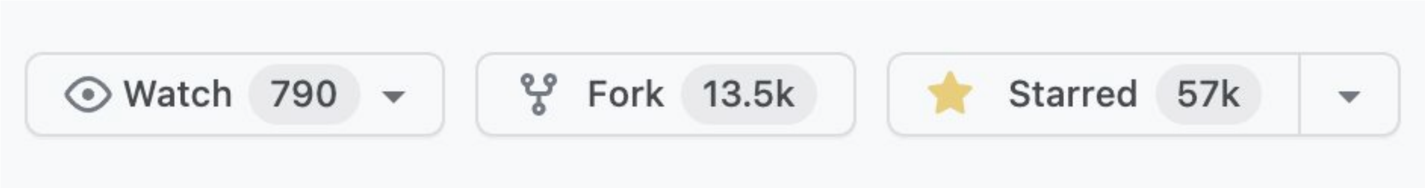
BERT-like architecture



BERT (Devlin et al. 2018)

Training transformers

huggingface / transformers



Transformers recipe

Thing that made our transformers work:

- small learning rate (1e-5)
- big batch size (at least 128 sentences)
  or use gradient accumulation
- freeze BERT encoder first and train only linear layer,
  then train jointly

BERTology works

LSTM	35.6
LSTM + BERT embeddings	46.0
DistillBERT	52.8
BERT-base	57.3

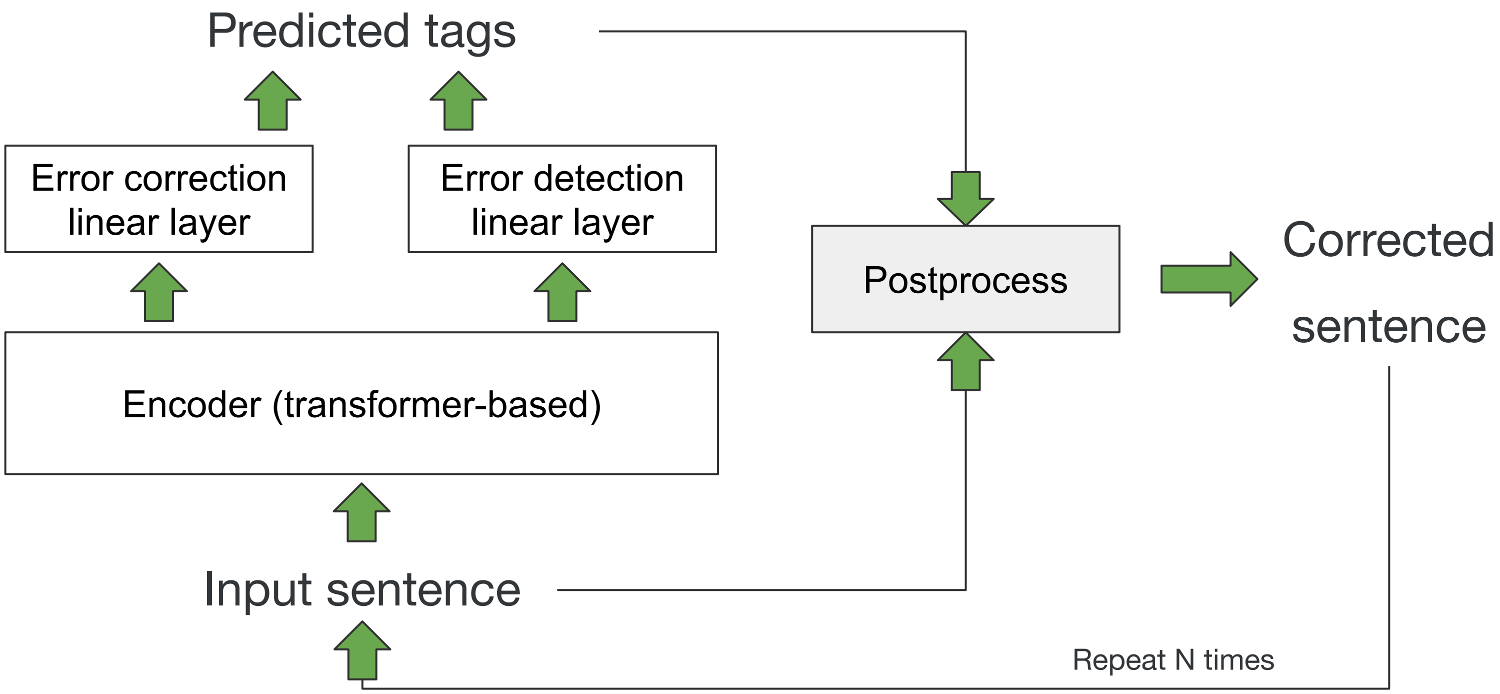
*on CoNLL-2014 (test)

Additional tricks

Iterative Approach

Sequence	A ten years old boy go school (zero corrections)
Iteration 1	A ten**-**years old boy **goes** school (2 total corrections)
Iteration 2	A ten-**year-**old boy goes **to** school (5 total corrections)
Iteration 3	A ten-year-old boy goes to school**.** (6 total corrections)

GECToR model: iterative pipeline



Staged training

- Training is splitted in N stages
- Each stage has its own data
- Each stage has its own hyperparameters
- On each stage model is initialized by the best weights of previous stage

Training stages

I. **Pre-training on synthetic errorful sentences as in (Awasthi et al., 2019)**
II. Fine-tuning on errorful-only sentences
III. Fine-tuning on subset of errorful and errorfree sentences as in (Kiyono et al., 2019)

Dataset	# sentences	% errorful sentences
PIE-synthetic	9,000,000	100.0%
Lang-8	947,344	52.5%
NUCLE	56,958	38.0%
FCE	34,490	62.4%
W&I+LOCNESS	34,304	67.3%

Training stages

I. Pre-training on synthetic errorful sentences as in (Awasthi et al., 2019)
II. **Fine-tuning on errorful-only sentences** (new!)
III. Fine-tuning on subset of errorful and errorfree sentences as in (Kiyono et al., 2019)

Dataset	# sentences	% errorful sentences
PIE-synthetic	9,000,000	100.0%
Lang-8	947,344	52.5%
NUCLE	56,958	38.0%
FCE	34,490	62.4%
W&I+LOCNESS	34,304	67.3%

Training stages

I. Pre-training on synthetic errorful sentences as in (Awasthi et al., 2019)
II. Fine-tuning on errorful-only sentences
III. **Fine-tuning on subset of errorful and errorfree sentences as in (Kiyono et al., 2019)**

Dataset	# sentences	% errorful sentences
PIE-synthetic	9,000,000	100.0%
Lang-8	947,344	52.5%
NUCLE	56,958	38.0%
FCE	34,490	62.4%
W&I+LOCNESS	34,304	67.3%

Training details

- Adam optimizer (Kingma and Ba, 2015);
- Early stopping after 3 epochs of 10K updates each w/a improvement;
- Epochs & batch sizes:
  - Stage I (pretraining): batch size=256, 20 epochs
  - Stages II, III (finetuning): batch size=128, 2-3 epochs

Training stage #	CoNLL-2014 (test)*, $F_{0.5}$	BEA-2019 (dev)*, $F_{0.5}$
I	49.9	33.2
II	59.7	44.4
III	62.5	50.3
III + Inf. tweaks	**65.3**	**55.5**

* Results are given for GECToR (XLNet).

Inference tweaks 1

By increasing probability of $KEEP tag we can force model to make only confident actions.

In such a way, we can increase precision by trading recall.

Inference tweaks 2

We also compute the minimum probability of incorrect class across all tokens in the sentence.

This value (min_erorr_probability) should be higher than threshold in order to run next iteration.

BERT family

Varying encoders from pretrained transformers

Encoder	CONLL-2014 (test)			BEA-2019 (test)		
	P	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**
LSTM**	51.6	15.3	35.0	-	-	-
ALBERT	59.5	31.0	50.3	43.8	22.3	36.7
BERT	65.6	36.9	56.8	48.3	29.0	42.6
GPT-2	61.0	6.3	22.2	44.5	5.0	17.2
RoBERTa	**67.5**	38.3	**58.6**	**50.3**	30.5	**44.5**
XLNet	64.6	**42.6**	58.6	47.1	**34.2**	43.8

* Training was performed on data from training stage II only. ** Baseline.

Varying encoders from pretrained transformers

Encoder	CONLL-2014 (test)			BEA-2019 (test)		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
LSTM**	51.6	15.3	35.0	-	-	-
ALBERT	59.5	31.0	50.3	43.8	22.3	36.7
BERT	65.6	36.9	56.8	48.3	29.0	42.6
GPT-2	61.0	6.3	22.2	44.5	5.0	17.2
RoBERTa	**67.5**	38.3	**58.6**	**50.3**	30.5	**44.5**
XLNet	64.6	**42.6**	58.6	47.1	**34.2**	43.8

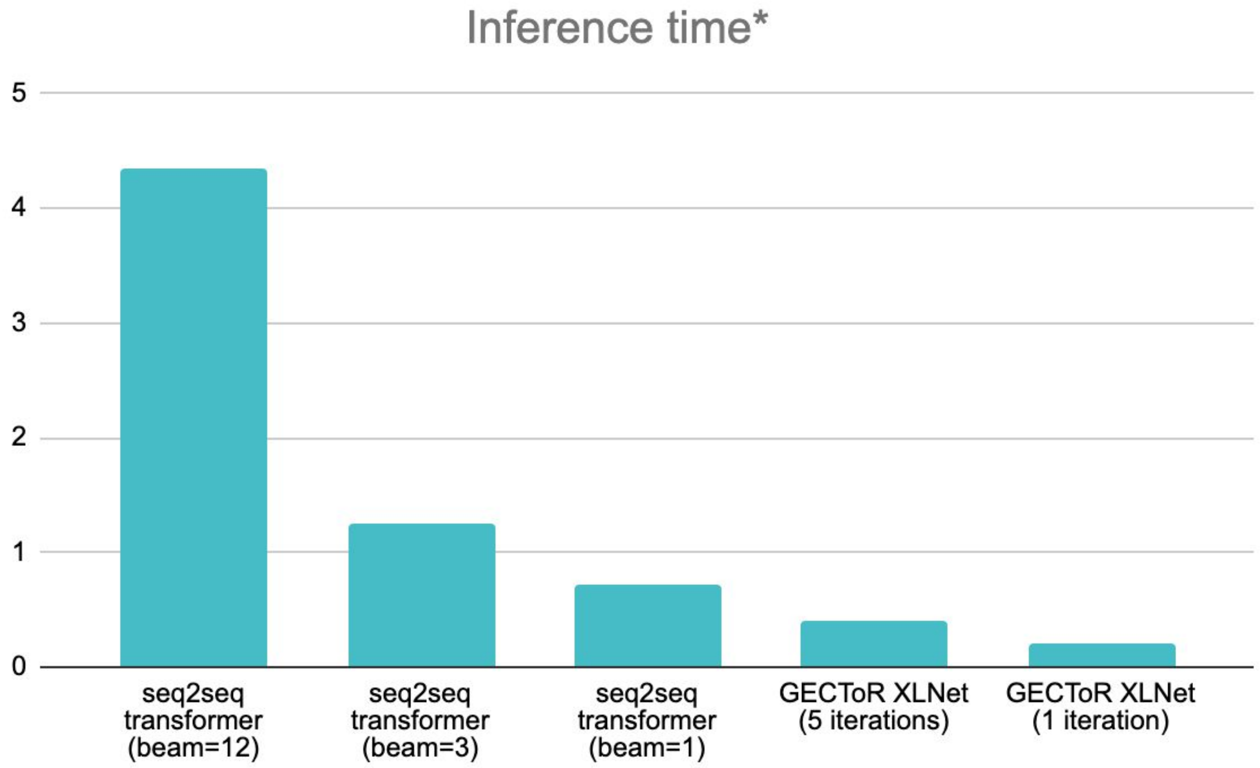
* Training was performed on data from training stage II only. ** Baseline.

Model inference time

Speed comparison

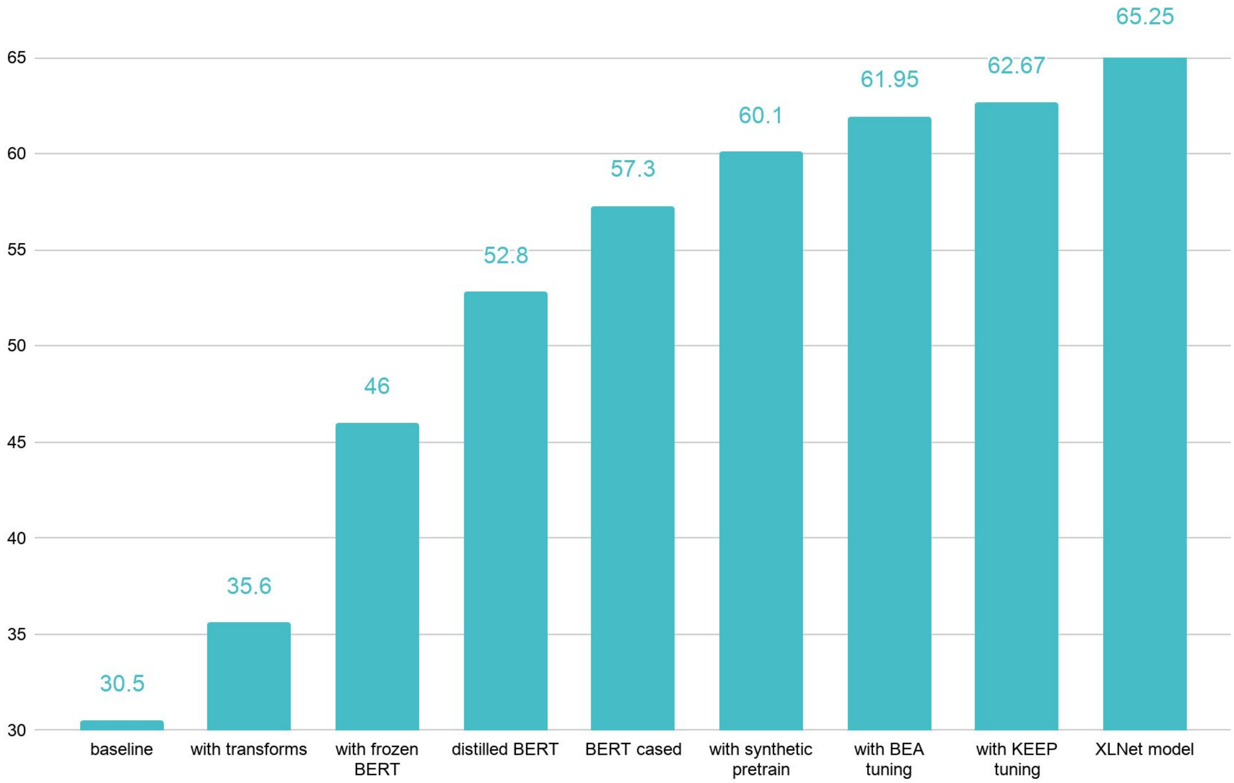
- NVIDIA Tesla V100
- CoNLL-2014 (test)
- single model
- batch size=128

Speed comparison



Final results

Big picture


65.25
62.67
61.95
60.1
57.3
52.8
46
35.6
30.5
65
60
55
50
45
40
35
30
baseline
with transforms
with frozen BERT
distilled BERT
BERT cased
with synthetic pretrain
with BEA tuning
with KEEP tuning
XLNet model

Results [2019]

GEC system	Ens.	CoNLL-2014 (test)			BEA-2019 (test)		
		P	R	$F_{0.5}$	P	R	$F_{0.5}$
Zhao et al. (2019)		67.7	40.6	59.8	-	-	-
Awasthi et al. (2019)		66.1	43.0	59.7	-	-	-
Kiyono et al. (2019)		67.9	**44.1**	61.3	65.5	**59.4**	64.2
Zhao et al. (2019)	✓	74.1	36.3	61.3	-	-	-
Awasthi et al. (2019)	✓	68.3	43.2	61.2	-	-	-
Kiyono et al. (2019)	✓	72.4	**46.1**	65.0	74.7	56.7	70.2
Kantor et al. (2019)	✓	-	-	-	78.3	58.0	73.2
GECToR (BERT)		72.1	42.0	63.0	71.5	55.7	67.6
GECToR (RoBERTa)		73.9	41.5	64.0	77.2	55.1	71.5
GECToR (XLNet)		**77.5**	40.1	**65.3**	**79.2**	53.9	**72.4**
GECToR (RoBERTa + XLNet)	✓	76.6	42.3	66.0	**79.4**	57.2	**73.7**
GECToR (BERT + RoBERTa + XLNet)	✓	**78.2**	41.5	**66.5**	78.9	**58.2**	73.6

Results [2022; CoNLL-2014 (test)]

Model	F0.5	Paper / Source	Code
T5 (t5.1.1.xxl) trained on cLang-8 (Rothe et al., ACL-IJCNLP 2021)	68.87	A Simple Recipe for Multilingual Grammatical Error Correction	T5, cLang-8
Tagged corruptions - ensemble (Stahlberg and Kumar, 2021)	68.3	Synthetic Data Generation for Grammatical Error Correction with Tagged Corruption Models	Official
Sequence tagging + token-level transformations + two-stage fine-tuning + (BERT, RoBERTa, XLNet), ensemble (Omelianchuk et al., BEA 2020)	66.5	GECToR – Grammatical Error Correction: Tag, Not Rewrite	Official

Source: nlpprogress.com [link]

Results [2022; BEA-2019 (test)]

Model	F0.5	Paper / Source	Code
GECToR large without synthetic pre-training - ensemble (Tarnavskyi and Omelianchuk, 2021)	76.05	Improving Sequence Tagging for Grammatical Error Correction	Official
T5 (t5.1.1.xxl) trained on cLang-8 (Rothe et al., ACL-IJCNLP 2021)	75.88	A Simple Recipe for Multilingual Grammatical Error Correction	T5, cLang-8
Tagged corruptions - ensemble (Stahlberg and Kumar, 2021)	74.9	Synthetic Data Generation for Grammatical Error Correction with Tagged Corruption Models	Official
Sequence tagging + token-level transformations + two-stage fine-tuning + (BERT, RoBERTa, XLNet), ensemble (Omelianchuk et al., BEA 2020)	73.6	GECToR – Grammatical Error Correction: Tag, Not Rewrite	Official

Source: nlpprogress.com [link]

Text Simplification

This work was done in 2020

This part of the presentation is based on the Text Simplification by Tagging paper written by:

- Kostiantyn Omelianchuk
- Oleksandr Skurzhanskyi
- Vipul Raheja

Text Simplification

- consists of modifying the content and structure of a text in order to make it easier to read and understand, while preserving its main idea and approximating its original meaning
- could benefit low literacy readers, English learners, children, and people with reading disabilities
- most commonly-used automatic metrics are:
  - SARI
  - FKGL
  - BLEU

Text Simplification

	Source	Target
Simplification	All students when attending a university must adhere to these guidelines.	All students when **going into** a university must **follow** these rules.
GEC	She see Tom is catched by policeman in park at last night.	She **saw** Tom **caught** by **a** policeman in **the** park last night.

Challenges

- Training data: only 2 publicly available training datasets:
    - WikiLarge (300k) is a set of automatically aligned complex-simple sentence pairs from English Wikipedia
    - Newsela includes thousands of news articles professionally leveled to different reading complexities. Has legal constraints to use it for public research.
- Evaluation: unreliable metrics (SARI, FKGL), like for almost every text generation task
- Adapt GECToR approach from GEC to Text Simplification

TST: GECToR for Simplification

- Edit-Tag Vocabulary
  - Tags overlap between tasks 92% allow to use GEC tags
- Data Preprocessing
  - Tried special preprocessing for Simplification task which was beneficial
- GEC Pretraining
  - Explored GEC initialization for Text Simplification
  - GECToR codebase is outdated (transformers 2.*) -> updated the code
  - Tokenization in GECToR was incorrect -> fixed
- Data Augmentations (details in next slide)
- Tagging models
  - Used RoBERTa-BASE for Text Simplification (vs. an ensemble of BERT, XLNET and RoBERTa used by GECToR)
- Inference Tweaks
  - $DELETE is highly important tag -> designed a new inference tweak for it

TST: Data Augmentations

- Standard Train/Test sets for Text Simplification
- **WikiAll**: 384k pairs collected from English Wikipedia-Simple Wikipedia
  - WikiSmall (88k Pairs)
  - WikiLarge (296k Pairs)
- **WikiBT**: Back-translated WikiAll (en-de and en-fr)
- **WikiEns**: Ensemble Distillation of 3 models
  - TST on WikiAll (randomly initialized)
  - TST-GEC on WikiAll (TST fine tuned on GEC task)
  - TST on WikiAll + WikiBT
- Final training set: **WikiAll** (384k Pairs) **+ WikiEns** (384k Pairs)

Results: Evaluation Metrics

- Metrics depend a lot on tokenization. 1+ points could be achieved by simply changing tokenization method
- SARI is calculated differently for the corpus-level:
  - It’s not just averaged of the sentence-level scores: statistics should be gathered on
    the whole corpus, then calculated
  - F1 is used for deletion operation
- BLEU is bad for the Text Simplification evaluation
- EASSE is a great evaluation package

Results: Simplification Quality

SOTA on WikiSmall, near-SOTA on TurkCorpus and ASSET

TurkCorpus

	SARI↑	ADD↑	DELETE↑	KEEP↑
Recent Works				
Xu et al. (2016b)	39.96	5.96	41.42	72.52
Nisioi et al. (2017)	35.66	2.99	28.96	**75.02**
Zhang and Lapata (2017)	37.27	-	-	-
Alva-Manchego et al. (2017)‡	37.08	2.94	43.20	65.10
Vu et al. (2018)	36.88	-	-	-
Zhao et al. (2018a)	40.42	5.72	42.23	73.41
Guo et al. (2018)	37.45	-	-	-
Qiang (2018)	37.21	-	-	-
Surya et al. (2019)	34.96	-	-	-
Dong et al. (2019)	38.22	3.36	39.15	72.13
Zhao et al. (2020b)	37.25	2.87	40.06	68.82
Mallinson et al. (2020)	38.13	3.55	40.45	70.39
Martin et al. (2020a)	41.38	-	-	-
Martin et al. (2020b)	**42.53**$_{\pm 0.36}$	-	-	-
Reference Baseline	40.02$_{\pm 0.72}$	6.21$_{\pm 0.60}$	**70.15**$_{\pm 1.35}$	43.69$_{\pm 1.46}$
Our System				
TST-BASE	39.17$_{\pm 0.77}$	3.62$_{\pm 0.41}$	41.61$_{\pm 3.14}$	72.29$_{\pm 1.45}$
TST-FINAL	41.46$_{\pm 0.32}$	**6.96**$_{\pm 0.44}$	47.87$_{\pm 0.75}$	69.56$_{\pm 1.19}$

‡ Quoted from the re-implementation by Dong et al. (2019).

ASSET

	SARI↑	ADD↑	DELETE↑	KEEP↑
Recent Works				
Martin et al. (2020a)	40.13	-	-	-
Martin et al. (2020b)	44.15$_{\pm 0.6}$	-	-	-
Reference Baseline	**44.89**$_{\pm 0.90}$	**10.17**$_{\pm 1.20}$	58.76$_{\pm 2.24}$	**65.73**$_{\pm 2.03}$
Our System				
TST-BASE	37.4$_{\pm 1.62}$	3.62$_{\pm 0.59}$	47.22$_{\pm 4.5}$	61.37$_{\pm 0.52}$
TST-FINAL	43.21$_{\pm 0.3}$	8.04$_{\pm 0.29}$	**64.25**$_{\pm 1.22}$	57.35$_{\pm 1.68}$

WikiSmall

	SARI↑	ADD↑	DELETE↑	KEEP↑
Recent Works				
Zhang and Lapata (2017)	27.24	-	-	-
Alva-Manchego et al. (2017)‡	30.50	2.72	76.31	12.46
Vu et al. (2018)	29.75	-	-	-
Guo et al. (2018)	28.24	-	-	-
Qiang (2018)	26.49	-	-	-
Dong et al. (2019)	32.35	2.24	**81.30**	13.54
Zhao et al. (2020b)	36.92	2.04	72.79	35.93
Reference Baseline	-	-	-	-
Our System				
TST-BASE	43.11$_{\pm 1.87}$	4.66$_{\pm 1.31}$	61.13$_{\pm 4.73}$	**63.54**$_{\pm 2.75}$
TST-FINAL	**44.67**$_{\pm 1.26}$	**8.12**$_{\pm 0.92}$	64.87$_{\pm 2.09}$	61.01$_{\pm 1.76}$

‡ Quoted from the re-implementation by Dong et al. (2019).

Results: Readability

SOTA on ASSET, Near-SOTA on TurkCorpus and WikiSmall

TurkCorpus

	FKGL↓
Recent Works	
Xu et al. (2016b)	7.29
Nisioi et al. (2017)	8.42
Zhang and Lapata (2017)	6.62
Alva-Manchego et al. (2017)‡	**5.35**
Vu et al. (2018)	-
Zhao et al. (2018a)	7.79
Guo et al. (2018)	7.41
Qiang (2018)	6.56
Surya et al. (2019)	-
Dong et al. (2019)	7.3
Zhao et al. (2020b)	-
Mallinson et al. (2020)	8.98
Martin et al. (2020a)	7.29
Martin et al. (2020b)	$7.60_{\pm 1.06}$
Reference Baseline	$8.77_{\pm 0.19}$
Our System	
TST-BASE	$8.08_{\pm 0.31}$
TST-FINAL	$7.87_{\pm 0.19}$

ASSET

	FKGL↓
Recent Works	
Martin et al. (2020a)	7.29
Martin et al. (2020b)	$7.60_{\pm 1.06}$
Reference Baseline	$\mathbf{6.49}_{\pm 0.42}$
Our System	
TST-BASE	$8.08_{\pm 0.31}$
TST-FINAL	$6.87_{\pm 0.27}$

WikiSmall

	FKGL↓
Recent Works	
Zhang and Lapata (2017)	7.55
Alva-Manchego et al. (2017)‡	9.38
Vu et al. (2018)	-
Guo et al. (2018)	6.93
Qiang (2018)	10.75
Dong et al. (2019)	**5.47**
Zhao et al. (2020b)	-
Reference Baseline	8.74
Our System	
TST-BASE	$8.41_{\pm 1.01}$
TST-FINAL	$9.29_{\pm 0.9}$

Results: Ablation Study

All adaptation steps progressively enhance the system

System	SARI ↑	FKGL ↓
TST	38.3 ± 1.36	8.08 ± 0.31
+ GEC	38.4 ± 0.83	8.32 ± 0.26
+ Filtering	39.1 ± 0.48	7.66 ± 0.25
+ WikiBT	39.5 ± 0.01	7.5 ± 0.06
+ WikiEns (- WikiBT)	40.3 ± 0.15	$\mathbf{7.48 \pm 0.2}$
+ InfTweaks	$\mathbf{42.3 \pm 0.25}$	7.87 ± 0.19

Average SARI and FKGL scores

Text Simplification by Tagging

Kostiantyn Omelianchuk, Vipul Raheja, Oleksandr Skurzhanskyi

Text Simplification

Rewriting text into a form that is easier to read and understand while preserving its underlying meaning and information.

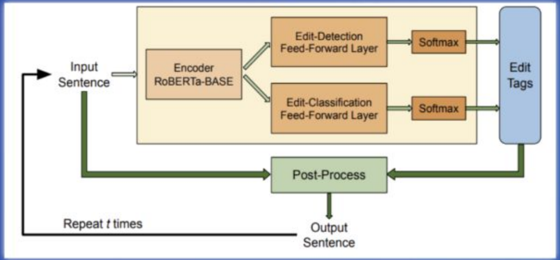
Input: Hinterrhein is an administrative district in the canton of Graubunden, Switzerland.

Output: Hinterrhein is a district of the canton of Graubunden, Switzerland.

Limitations of existing approaches

- Limited interpretability and insight into simplification operations
- Little control or adaptability to different aspects of simplification
- Not sample-efficient
- Slow inference speeds owing to autoregressive decoding

Our Approach



1. Define **custom edit transformations** (token-level edit tags)
2. Perform **iterative sequence tagging** to convert target sequences to tag sequences
3. **Fine-tune** pre-trained transformers to predict the tag sequences

Edit Transformations

KEEP	keep current token unchanged	DELETE	delete current token
APPEND_t_i	append new token t_i	REPLACE_t_i	replace current token with token t_i
CASE	change the case of current token	MERGE	merge current and next tokens
NOUN NUMBER	convert singular noun to plurals and vice versa	VERB FORM	change the form of verbs

Systemic Enhancements

We propose several approaches to improve the performance of the model on the task:

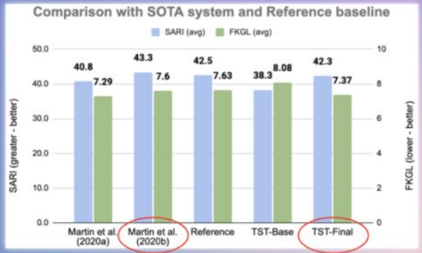
- Initialize the model with weights trained for GEC
- Filter out bracketed text
- Enriching data with back-translated data (**WikiBT**);
- Enriching data with ensemble-generated data (**WikiEns**)
- Tune confidence biases to adjust the probabilities of `KEEP` and `DELETE` edit-tags (**InfTweaks**)

Ablation Study

System	SARI ↑	FKGL ↓
TST	38.3 ± 1.36	8.08 ± 0.31
+ GEC	38.4 ± 0.83	8.32 ± 0.26
+ Filtering	39.1 ± 0.48	7.66 ± 0.25
+ WikiBT	39.5 ± 0.01	7.5 ± 0.06
+ WikiEns (- WikiBT)	40.3 ± 0.15	**7.48 ± 0.2**
+ InfTweaks	**42.3 ± 0.25**	7.87 ± 0.19

Average SARI and FKGL scores (ASSET and TurkCorpus test set)

Results



Inference Speed

System	Inference time (sec)
BART, beam size = 8	2.82
BART, beam size = 2	1.95
ACCESS, beam size = 8	1.43
ACCESS, beam size = 1	1.14
TST, 5 iterations	0.43
TST, 4 iterations	0.39
TST, 3 iterations	0.33
TST, 2 iterations	0.24
TST, 1 iteration	0.13

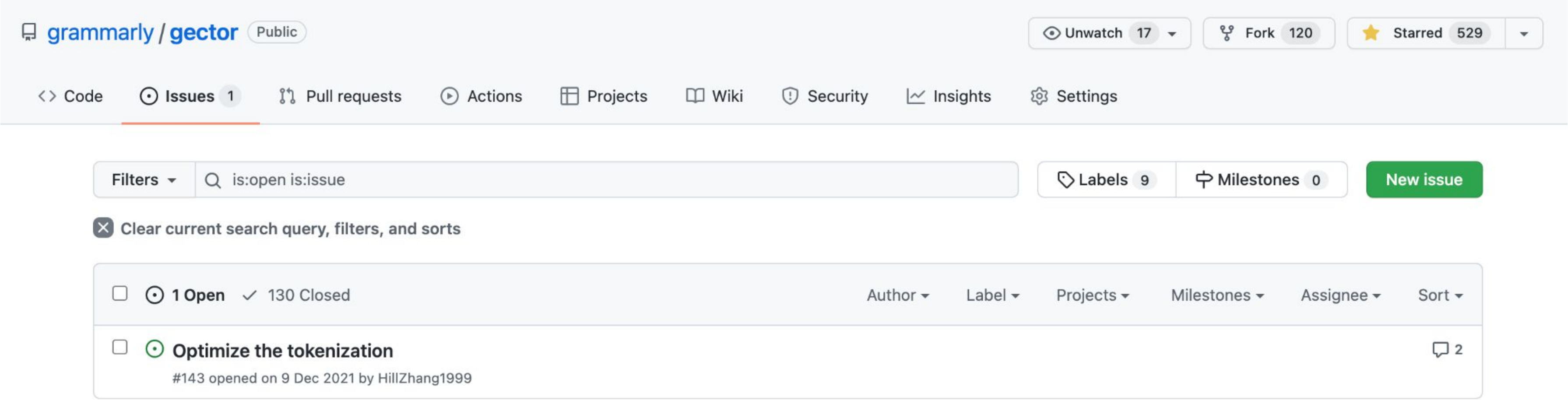
Average inference time per batch

Conclusions

- We present TST, a simple and efficient Text Simplification system based on **sequence tagging** and **pre-trained transformers**
- TST is highly interpretable as it provides detailed insights into simplification operations.
- TST achieves **near-SOTA results** with TST while being **11x faster** than previous SOTA
- Using proposed **data augmentations** and **inference tweaks** leads to substantial improvements on the task

Reusable artifacts

Repository & models



Repository & models

Pretrained models

Pretrained encoder	Confidence bias	Min error prob	CoNNL-2014 (test)	BEA-2019 (test)
BERT [link]	0.1	0.41	61.0	68.0
RoBERTa [link]	0.2	0.5	64.0	71.8
XLNet [link]	0.2	0.5	63.2	71.2

Note: The scores in the table are different from the paper's ones, as the later version of transformers is used. To reproduce the results reported in the paper, use this version of the repository.

	SARI		FKGL
Model	TurkCorpus	ASSET	
TST-FINAL [link]	39.9	40.3	7.65
TST-FINAL + tweaks	41.0	42.7	7.61

Links

- GEC paper
- Text Simplification paper
- The code

Questions?

grammarly