GEC Review

Nuri Tas

August 2023

1 Grammatical Error Correction

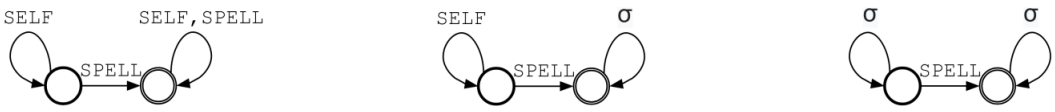
Grammatical error correction (GEC) aims to systematize writing errors to correct grammar and spelling errors. However, one of the biggest bottlenecks in GEC is data sparsity Stahlberg and Kumar [Stahlberg and Kumar], namely the need for more diverse and large exemplary data. Although there have been many attempts to generate synthetic data based on various models, such as heuristic-based random words, such models failed to represent the proper distribution of grammatical errors in the wild Stahlberg and Kumar [Stahlberg and Kumar]. We will explain tagged corruption models by Stahlberg and Kumar [2021] that enables a more flexible distribution of synthetic data and produces state-of-the-art results on GEC baselines.

1.1 Tagged Corruption Models

This section summarizes the paper Stahlberg and Kumar [2021]

Tagged corruption models create an ungrammatical (corrupt) sentence from a clean sentence based on an error tag $\tau \in T$, where T denotes 25 error tags supported by ERRANT.

The model is trained using Seq2Edits Stahlberg and Kumar [2020], which can predict the error tags together with the edits. Additionally, Finite State Transducers (FST) are utilized to force to generate a particular error tag. The following figure shows the FSTs for the $SPELL$ tag.



(a) NoSigma (b) PostSigma (c) PrePostSigma

Figure 1: Image credit: Stahlberg and Kumar [2021]

In Fig 1, σ refers to any error tag and $SELF$ denotes the source spans that are not modified, i.e., the change points where corruption is applied. In Fig 1a,

only includes $SELF$ and $SPELL$, whereas Fig 1b can have any error tag after starting with $SELF$ and $SPELL$ tags. Fig 1c, however, can start and end with any error tags as long as including a $SPELL$ tag in between.

1.1.1 Synthetic Data Generation with Tagged Corruption Models

Let x_n ($n \in [1, N]$) and $y_{t,n}$ denote the input sentence and the corresponding corrupted sentence for a given tag $\tau \in T$, respectively. One of the goals is to only apply one tag per sentence and create a new corrupted corpus from the input data such that the distribution of the error tags are equal:

$$\forall \tau \in T : P(\tau) \approx= \frac{|\{\tau_n = \tau | n \in [1, N]|\}}{N} \tag{1}$$

Three methods are compared: Offline-Optimal, Offline-Probabilistic, Online

Offline-Optimal The offline-optimal model searches for the tag with the highest log-probability under the constraint that the distribution of the tags matches the desired distribution:

$$\max_{\tau^*} \sum_{n=1}^{N} log P(y_{\tau_n^*,n} | \tau_n^*, x_n) \tag{2}$$

It can be shown that this model can be solved with the minimum-cost solver.

Offline-Probabilistic In the offline-probabilistic model, the samples are created according to a given distribution, and then N sentences with the highest likelihood to contain a given tag is drawed:

$$P((x, y)|\tau) = \frac{P(\tau|(x, y)P(x, y)}{\sum_{n=1}^{N} P(\tau|(x_n, y_n))P(x_n, y_n)} \tag{3}$$

assuming each corrupted sentence sample for a given tag has equal probability, i.e., $P(x, y) = \frac{1}{N}$, we obtain

$$P((x, y)|\tau) = \frac{P(\tau|(x, y)}{\sum_{n=1}^{N} P(\tau|(x_n, y_n))} \tag{4}$$

Now,

$$P(\tau|(x, y)) = \frac{P(\tau, x, y)}{P(x, y)} = \frac{P(x)P(\tau|x)P(y|(\tau, x))}{P(x, y)} = \frac{\frac{1}{N}\frac{1}{|T|}P(y|(\tau, x))}{\frac{1}{N}} = \frac{1}{|T|}P(y|(\tau, x)) \tag{5}$$

Note that, since we draw the corrupted sentences with the highest probability to contain a given tag, the model may include corrupted sentences derived from the same sentence or exclude some sentences in the input data. This is not an issue in the offline-optimal model as the model already takes each input sentence as a supply node (in the minimum-cost problem terminology) and derive the optimal tag accordingly.

Online Both offline-optimal and offline-probabilistic models have to create $N * |T|$ corrupted sentences to find the optimal solutions, and hence each have $\Omega(N|T|)$ time-complexity. This issue can be optimized by assigning tags on the fly according to a given distribution and creating a model with $\Omega(N)$ time complexity.

1.2 ERRANT for Turkish

As a first attempt, we give corresponding grammatical errors according to ERRANT error tags. Note that some tags don't have a Turkish example, i.e., CONTR (n't $\rightarrow$ not) or DET (the $\rightarrow$ a). The following table is taken from Bryant [2019] and adjusted for Turkish.

Code	Meaning	Description\Example
ADJ	Adjective	büyük adam oldun → koca adam oldun
ADJ:FORM	Adjective Form	→ Comparative or superlative adjective errors. Since the corresponding Turkish comparative and superlative adjectives (daha/en) are strict, it is unlikely to encounter errors here.
ADV	Adverb	çabukca → çabucak
CONJ	Conjunction	ve → ama
CONTR	Contraction	-
DET	Determiner	-
MORPH	Morphology	This can be tricky in Turkish. For instance, both hızlı (adj) and hızlıca (adv) can be used interchangeably: hızlı geldin ∼ hızlıca geldin
NOUN	Noun	kafam ağrıyor → başım ağrıyor
NOUN:INFL	Noun Inflection	We adjust the original definition to the correct use of "ler" or "lar" for plural words. Defterlar → defterler
NOUN:NUM	Noun Number	iki kişiydi → iki kişilerdi
NOUN:POSS	Noun Possessive	-
ORTH	Orthography	birşey → bir şey
OTHER	Other	Unclassified errors; e.g. paraphrases.
PART	Particle	-
PREP	Preposition	sen de kalalım → sende kalalım
PRON	Pronoun	bizimki çocuklar → bizim çocuklar
PUNCT	Punctuation	; → ,
SPELL	Spelling	entellektüel → entelektüel
UNK	Unknown	Detected but not corrected errors
VERB	Verb	geldim → gittim
VERB:FORM	Verb Form	gitme için bekledik → gitmek için bekledik
VERB:INFL	Verb Inflection	Misapplication of tense morphology. geldem → geldim
VERB:SVA	Subject-Verb Agreement	(Ben) geldin → (Ben) geldim
VERB:TENSE	Verb Tense	
WO	Word Order	hemen çocuk geldi → çocuk hemen geldi. Note: Turkish is more malleable against the word order.

2 GECTurk Paper Summary

The paper Kara et al. [2023] introduces an algorithm to corrupt grammatically correct sentences based on 25 error types created by the authors, where the transformation and their inverses are denoted as f and f^{-1}, respectively. For a given sentence, the algorithm first shuffles fs , after which each f has these inputs: a given sentence s, morphological analysis of the sentence M_s by the morphological analyzer from Dayanik et al. [2018], whether any transformation applied previously by other fs - named $flags$, and the probability p to control the frequency of error types.

Probabilities of transformations are proportional to the frequency of error types made by native Turkish speakers. A transformation iterates over tokens of a sentence and may modify the sentence only if no other transformation is made to the respective token. Note that as transformations f are clearly formulated, we also have their inverses, which will become very handy in correcting corrupted sentences. The paper also presents a corpora of 138K unique sentences on previously compiled corpora (Diri and Amasyali, 2003; Amasyalı and Diri, 2006; Can and Amasyalı, 2016; Kemik NLP Group, 2022) where only half of the sentences are modified by the given algorithm to let models learn the correct grammatical structures of Turkish as well. The full list of transformation and their annotations/descriptions are given in the Figure 2. Note that, applying the inverse transformation to a corrupted sentence does not necessarily yield a grammatically correct sentence.

Additionally, a curated dataset of 300 movie reviews named MovieReview are introduced by the paper, only half of which are again corrupted.

2.1 Models

Three models are used to evaluate the performance of GECTurk:

1. Neural Machine Translation (NMT) Baseline: A Vanilla Transformer model with the training dataset consisting of $(x_i, y_i, a_i)_i^N$ triplets where x_i and y_i are the grammatically incorrect and correct sentences, respectively, and a_i are the annotations corresponding to each error type, if any, in the given sentence. NMT is used only for GEC task.

2. Sequence Tagger: A cased BERT model pretrained on Turkish text with an additional linear and softmax layer. Sentences are tokenized by WordPiece tokenizer and for words with multiple subword tokens, only the first of those subword tokens are fed into the model. Softmax layer enables the model to yield the probabilities of various error types, hence this model can perform GED. For GEC task, the inverse of the function corresponding to an error type is utilized to get grammatically correct sentences.

3. Prefix Tuning: In prefix tuning, N artificial tokens are appended to each input. Hence for an input sentence $x = (x_1, \ldots, x_T)$, the modified input becomes $x' = (s_1, \ldots, s_N, x_1, \ldots, x_T)$ where $(s_1, \ldots, s_N)$ are the artificial

Category	Rule ID	Description	f	Frequency
-DE/-DA	1. CONJ_DE_SEP	Conjunction "-de/-da" is written separately.	Durumu [oğluna da -> oğlu nada] bildirdi.	12962
	2. CONJ_DE_VH	Conjunction "-de/-da" must follow the vowel harmony	Çok [da -> de] iyi olmuş.	101
	3. CONJ_DE_AR	Conjunction "-de/-da" does not follow phonetic assimilation rules.	Sınıf [da -> ta] temizlendi.	99
	4. YADA	"-de/-da" written together with the word "ya" is always written separately.	Sen [ya da -> yada] o buradan gidecek.	472
	5. CONJ_DE_APOS	Conjunction "-de/-da" cannot be used with an apostrophe.	[Ayşe de -> Ayşe'de] geldi.	10859
	6. CASE_DE	Suffix "-de/-da" is written adjacent.	[Evde -> Ev de] hiç süt kalmamıştı.	37462
-KI	7. CONJ_KI_SEP	Conjunction "-ki" is written separately.	Bugün öyle çok [yorulmuş ki -> yorulmuşki] hemen yattı.	1817
	8. CONJ_KI_EXC	On some exceptional instances, by convention, the conjunction "-ki" is written adjacent.	[Belki -> Bel ki], [oysaki -> oysa ki], [çünkü -> çünki]	1395
FOREIGN	9. FOREIGN_R1	Words that start with double consonants of foreign origin are written without adding an "-i" between the letters.	[gram -> giram]	307
	15. FOREIGN_R2	Some foreign origin words undergo consonant assimilation	[sebebi -> sebepi]	327
	16. FOREIGN_R2_EXC	Exceptions to the FOREIGN_R2 rule	[evrakı -> evrağı]	422
BISYL	13. BISYLL_HAPL._VOW	Some bisyllabic words undergo haplology when they get a suffix starting with a vowel.	[ağzı -> ağızı]	1567
	14. BISYLL_HAPL._VOW_EXC	Exception to previous rule	[içeride -> içerde]	866
LIGHT VERB	17. LIGHT_VERB_SEP	Light verbs such as "etmek, edilmek, eylemek, olmak, olunmak" are written separately in case of no phonological assimilation	[arz etmek -> arzetmek]	460
	18. LIGHT_VERB_ADJ	Light verbs are written adjacent in case of phonological assimilation e.g., liaison	[emretti -> emir etti]	547
COMPOUND	20. COMP_VERB_ADJ	Compound words formed by knowing, giving, staying, stopping, coming, and writing are written adjacent if they have a suffix starting with -a, -e, -ı, -i, -u, -ü.	[uyuyakalma -> uyuya kalmak], [gidedurmak -> gide durmak], [çıka gelmek -> çıka gelmek]	7840
SINGLE	22. PRONOUN_EXC	Traditionally, some pronouns are written adjacent.	[hiçbir -> hiç bir], [herhangi -> her hangi]	3867
	23. SENT_CAP	The first letter of the sentence is capitalized.	[Onlar -> onlar] geldi.	2613
	24. CAPPED	Some Arabic and Persian originated words are written with capped letters.	[kâğıt -> kağıt], [karargâh -> karargah]	834
	25. ABBREV	Grammatical rules for abbreviations, such as adding suffixes to abbreviations, punctuations with abbreviations etc.	[Alm. -> Alm], [THY'de -> THY'da], [cm'yi -> cm'yi]	359
	12. PRONOUNC_EXC	Unlike its pronunciation, verbs ending with "-a/-e" do not mutate when they get a suffix other than "-yor"	[başlayacağım -> başlıyacağım]	12750

Figure 2: Transformation Descriptions

soft tokens. During training, only the newly added tokens are optimized while the rest of the model is frozen. mGPT is utilized for sequence

generation with the prompt provided by OpenPrompt. The training set also needs to include the error types and their locations for each input. An example output of mGPT is given in the Figure 3 below.

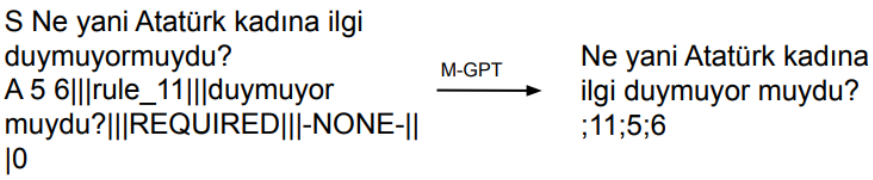


Figure 3: A sample mGPT output

2.2 Datasets and Evaluation

Datasets: The models are evaluated on GECTurk, MovieReview datasets, which were explained previously, and on the BOUN Dataset Ugurcan Arikan and Uskudarli [2019]
Evaluation: For GEC, Precision (P), Recall (R), and $F_{0.5}$, and for GED, P, R, and F_1 scores are calculated, respectively.

2.3 Experiment Results

The experiment results are given in the Figure 4. Curated Test Data in the figure corresponds to the zero-shot results on the out-of-domain MovieReview dataset. It can be clearly seen that both SeqTag and mGPT outperforms the baseline model for GEC task, where they achive 0.98 and 0.94 $F_{0.5}$ scores, respectively. However, mGPT lacks quite behind SeqTag for GED task, where SeqTaq still realizes state-of-the-art results with 0.9 F_1 score.

Note that mGPT is a generative model and predicts both the next token and detection on the fly, so there is not a strong correlation between detection and corrections, which can also be confirmed from the experiment results. On the other hand, SeqTag follows a pipeline approach wherein it first detects the errors and apply inverse transformations to correct sentences.

	GECTurk					
	Detection			Correction		
	P	**R**	F_1	**P**	**R**	$\mathbf{F_{0.5}}$
NMT	-	-	-	0.50 ± 0.01	0.84 ± 0.01	0.55 ± 0.01
SeqTag	0.90 ± 0.003	0.90 ± 0.013	0.90 ± 0.006	0.98 ± 0.001	0.98 ± 0.001	0.98 ± 0.001
mGPT	0.52 ± 0.02	0.38 ± 0.004	0.41 ± 0.01	0.95 ± 0.01	0.92 ± 0.03	0.94 ± 0.01
Curated Test Data						
NMT	-	-	-	0.31	0.62	0.35
SeqTag	0.94	0.87	0.89	0.85	0.80	0.84
mGPT	0.73	0.52	0.59	0.75	0.61	0.72

Figure 4: Experiment Results

To investigate the knowledge transfer ability of the models, the experiment results on the BOUN dataset are also given in Figure **??**. Here the complex split is 100 sentences on the BOUN dataset that are mentioned to be challenging by the authors Ugurcan Arikan and Uskudarli [2019]. Additionally, models are trained from scratch on the BOUN dataset and their results are also given. However, NMT turned out to fail to converge and its results are shown as 0.

The results show that SeqTag surpasses the BOUN results by 0.4 pp on F_1 score when it is trained from the scratch. It is also noteworthy that mGPT yields similar scores compared to zero-shot detection even when it is trained on the BOUN dataset, which can be interpreted that i) GECTurk feeds model high quality prior knowledge ii) pretrained models have a larger knowledge transfer ability

References

Bryant, C. (2019, June). Automatic annotation of error types for grammatical error correction. Technical Report UCAM-CL-TR-938, University of Cambridge, Computer Laboratory.

Dayanik, E., E. Akyürek, and D. Yuret (2018). Morphnet: A sequence-to-sequence model that combines morphological analysis and disambiguation.

Kara, A., F. Marouf Safian, A. Bond, and G. Gül Sahin (2023). Gecturk: Grammatical error correction and detection dataset for turkish.

Stahlberg, F. and S. Kumar. The C4_200M Synthetic Dataset for Grammatical Error Correction.

Stahlberg, F. and S. Kumar (2020, November). Seq2Edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, pp. 5147–5159. Association for Computational Linguistics.

Stahlberg, F. and S. Kumar (2021, April). Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, Online, pp. 37–47. Association for Computational Linguistics.

Ugurcan Arikan, O. G. and S. Uskudarli (2019). Detecting clitics related orthographic errors in turkish. In *In Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP 2019, Varna, Bulgaria, September 2-4, 2019, pages 71–76.*