ПРОГРАММИРОВАНИЕ, 2024, № 4, с. 41–48

ПРОГРАММНАЯ ИНЖЕНЕРИЯ, ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММ

УДК 004.89

RuGECToR: НЕЙРОСЕТЕВАЯ МОДЕЛЬ НА ОСНОВЕ ПРАВИЛ ДЛЯ ИСПРАВЛЕНИЯ ГРАММАТИЧЕСКИХ ОШИБОК НА РУССКОМ ЯЗЫКЕ

© 2024 г. И. А. Хабутдинов[a, b, *], А. В. Чащин[b, **], А. В. Грабовой[a, b, *], А. С. Кильдяков[b, ****], Ю. В. Чехович[b, c, *****]**

[a] *Московский физико-технический институт (национальный исследовательский университет)*
141700 Московская область, г. Долгопрудный, Институтский пер., 9, Россия
[b] *АО "Антиплагиат"*
117105 Москва, Варшавское шоссе, д. 33, эт. 14, комната 15в, Россия
[c] *Федеральный исследовательский центр "Информатика и управление" РАН*
119333 Москва, ул. Вавилова, д. 44, кор. 2, Россия
**E-mail: khabutdinov@ap-team.ru*
***E-mail: chashchin@ap-team.ru*
****E-mail: grabovoy@ap-team.ru*
*****E-mail: kildyakov@ap-team.ru*
******E-mail: chehovich@ap-team.ru*

Поступила в редакцию 21.02.2024
После доработки 18.03.2024
Принята к публикации 18.03.2024

Исправление грамматических ошибок является одной из основных задач обработки естественного языка. В настоящий момент наиболее эффективной моделью, использующей подход Sequence Tagging с открытым исходным кодом, для английского языка является модель GECToR. Для русского языка данная задача не имеет настолько эффективных решений ввиду отсутствия достаточного количества размеченных данных. Это послужило причиной проведения данного исследования. В исследовании описан процесс создания синтетического набора данных и обучения на нем модели. Архитектура GECToR адаптирована для русского языка и названа соответствующим образом – RuGECToR. Выбор архитектуры обусловлен тем, что в отличие от подхода Sequence-to-Sequence, она проста в интерпретации и не требует большого количества обучающих данных. Целью исследования было обучить модель таким образом, чтобы она обобщала морфологические свойства языка, а не подстраивалась под обучающую выборку. Представленная модель показала результат 82.5 на синтетических данных и 22.2 на наборе данных RULEC с точки зрения метрики $F_{0.5}$, при этом набор данных RULEC не использовался на этапе обучения.

Ключевые слова: обработка естественного языка, исправление грамматических ошибок, тегирование последовательности, генерация наборов данных, настройка параметров

DOI: 10.31857/S0132347424040048, **EDN:** PTEUNV

1. ВВЕДЕНИЕ

В современном мире проблема автоматического исправления грамматических ошибок (ИГО) является актуальной [1–3] в связи с увеличением объема текстовых данных [4]. Ручная проверка больших текстов является трудоемкой задачей. Кроме того, решение данной задачи имеет специфические приложения: проверка сочинений [5]; исправление сообщений в социальных сетях [6]; исправление текстов, написанных иностранными студентами [7]; поиск текстовых заимствований [8]. Последнее объясняется тем, что грамматические ошибки являются одним из способов обхода системы поиска текстовых заимствований [9]. Для обучения моделей поиска текстовых заимствований используются тексты научных статей, содержащие небольшое количество ошибок. Пользователи при этом могут нарочно допускать множество ошибок, влияя тем самым на предсказания модели. Это приводит к появлению состязательных атак (англ. adversarial attacks) на модели обработки естественного языка [10].

Большинство исследований в области ИГО ориентировано на английский язык [11]. Для многих других языков, включая русский, число исследований значительно меньше [12]. Особенно стоит отметить сложную морфологию русского языка, что усложняет решение задачи ИГО.

В настоящее время существуют два наиболее эффективных подхода решения задачи ИГО для английского языка, – Sequence-to-Sequence (Seq2Seq) [13, 14] и Sequence Tagging (ST) [15, 16]. В подходе Seq2Seq исходная последовательность представляет собой предложение с ошибками, а целевая последовательность – предложение без ошибок. Такой подход решения задачи ИГО работает хорошо, но обладают низкой скоростью работы, а также плохой интерпретируемостью, поскольку для определения типа ошибки необходимо использовать дополнительный функционал. Модели, основанные на подходе ST, не имеют данных проблем: не требуется полностью генерировать предложение без ошибок, достаточно лишь отметить ошибки в исходном предложении. Кроме того, они легко интерпретируемы, так как решают задачу классификации для каждого токена, сопоставляя ему нужное правило из заданного словаря корректирующих правил.

В [12] авторы сравнивают данные подходы для решения задачи ИГО в текстах на русском языке. Результаты показывают, что на небольшом объеме размеченных данных методы, основанные на подходе ST, существенно превосходят методы, основанные на подходе Seq2Seq. Для сравнения методов авторы представляют набор данных RULEC, содержащий предложения с размеченными грамматическими ошибками. Набор данных состоит из сочинений иностранных студентов.

Для решения задачи ИГО на английской языке наиболее эффективной ST-моделью является GECToR [15]. Данная модель состоит из кодировщика на основе архитектуры трансформер [17] и двухголового классификатора. Первая голова предсказывает наличие ошибки, а вторая – соответствующее правило для исправления ошибки. После этого соответствующее правило применяется к каждому токену последовательности для исправления. Если токен не содержит ошибок, то данному токену ставится в соответствие правило “$KEEP”, которое оставляет данный токен без изменений. Обучение модели GECToR состоит из трех этапов:

1) обучение на синтетических данных;

2) дообучение на корпусе данных, содержащем ошибки;

3) дообучение на комбинации из корпусов данных с ошибками и без ошибок.

В работе предложен алгоритм генерации синтетического набора данных, содержащего тексты на русском языке для первого этапа обучения. Модель GECToR адаптирована в модель RuGECToR для ИГО в русскоязычных текстах с использованием сгенерированных синтетических данных и данных из открытых источников. Данная модель не требует большого количества данных для обучения, в отличие от моделей на основе архитектуры Seq2Seq, и показывает конкурентноспособное качество для английского языка. Целью данного исследования является обучение модели, способной обобщать морфологические свойства русского языка, а не подстраиваться под обучающую выборку.

2. ПОСТАНОВКА ЗАДАЧИ ИСПРАВЛЕНИЯ ГРАММАТИЧЕСКИХ ОШИБОК

Задано множество пар, в котором каждая пара состоит из предложения s_i и соответствующего ему эталонного исправления t_i:

$$S = \left\{\left(s_i, t_i\right)\right\}_{i=1}^{N},$$

где N – количество пар. В данной работе предполагается, что каждое предложение представлено в виде последовательности токенов $s_i = \left\{x_1, x_2, \ldots, x_{n_i}\right\}$, а каждое эталонное исправление – в виде последовательности корректирующих правил $t_i = \left\{y_1, y_2, \ldots, y_{n_i}\right\}$, где n_i – длина i-го предложения. Под понятием “токен” подразумевается слово, таким образом разбиение предложения на токены происходит на уровне слов. Каждое корректирующее правило является элементом заданного нами словаря: $y_j \in t_i$, $j = 1, \ldots, n_i$. Составление словаря описано ниже.

Целью задачи является нахождение функции T для построения отображения из последовательности токенов s_i в последовательность корректирующих правил t_i:

$$T : s_i \mapsto t_i \in \left\{0, 1, \ldots, k\right\}^{n_i}$$

где k – размер словаря корректирующих правил.

3. ПРЕДЛАГАЕМЫЙ ПОДХОД ДЛЯ РЕШЕНИЯ ЗАДАЧИ ИСПРАВЛЕНИЯ ГРАММАТИЧЕСКИХ ОШИБОК

На вход модели GECToR подается предложение, которое требуется исправить. Далее предложение разбивается на последовательность

токенов. Таким образом, задача ИГО сводится к нахождению отображения T для сопоставления каждого токена с соответствующим правилом из словаря корректирующих правил. Для русского языка построен словарь, состоящий из 5183 правил, которые будут подробно описаны ниже.

Данная постановка задачи имеет недостаток: каждый токен сопоставляется только с одним правилом из словаря, но иногда для исправления ошибки требуется большее количество изменений. Для решения данной проблемы предлагается использовать итеративное исправление: предложение, полученное в результате работы модели, снова подается модели на вход. Таким образом, модель предсказывает правила для новой последовательности токенов. Правила разработаны таким образом, что последовательность токенов с ошибками может быть преобразована в последовательность токенов без ошибок за конечное число итераций. Проделаны следующие адаптации модели GECToR для русского языка:

1) проведено обучение модели RuGECToR в два этапа:
   a) обучение на синтетических данных с ошибками;
   b) дообучение на комбинации из синтетических данных с ошибками и без ошибок;
2) изменен этап применения модели для исправления ошибок с использованием библиотеки pymorphy2 [18];
3) составлен словарь правил для исправления ошибок;
4) в качестве кодировщика использован Multilingual BERT [19].

4. ОПИСАНИЕ НАБОРА ДАННЫХ

Для генерации синтетических ошибок на первом этапе обучения взяты предложения из русскоязычной Википедии [20] и школьных сочинений [21]. Для второго этапа обучения использованы школьные сочинения [21] и литературные тексты [22]. Сгенерированы специализированные ошибки для следующих частей речи: глагол, имя прилагательное, имя существительное, местоимение, причастие и числительное. Перед началом процесса генерации ошибок пронумерованы позиции токенов для указанных частей речи во всех предложениях.

Процесс генерации ошибок осуществляется следующим образом:

1) случайным образом выбирается предложение, где для каждого токена случайным образом генерируется ошибка, соответствующая части речи токена;
2) после этого каждому такому токену ставится в соответствие правило из словаря корректирующих правил, которое необходимо применить для исправления ошибки.

Распределение ошибок близко к равномерному. В табл. 1 показан пример генерации ошибок в предложении.

В табл. 2 показано, что для первого этапа обучения использовалось 10.000.000 предложений, где каждое предложение содержит произвольные типы ошибок кроме пунктуационных. Для второго этапа обучения мы использовали 1.000.000 предложений: 500.000 предложений не содержат ошибок; 250.000 предложений содержат все ошибки, кроме пунктуационных; оставшиеся 250.000 предложений содержат только пунктуационные ошибки. Для проведения второго этапа обучения добавлены литературные произведения и исключена Википедия с целью сделать модель более устойчивой к различиям между наборами данных [23]. В качестве тестовых наборов данных использовались школьные сочинения и тестовое подмножество набора данных RULEC.

В работе [15] авторы разделяют правила, применяемые к исходным токенам $\{x_1, x_2, \ldots, x_{n_i}\}$ для получения целевого предложения, на два типа – базовые и грамматические. В нашем исследовании сохранено данное разделение. Грамматические правила подобраны таким образом, чтобы словарь корректирующих правил покрывал множество правил русского языка. Базовые правила выполняют наиболее распространенные операции редактирования на уровне токенов: сохранение текущего токена x_i без изменений – правило $KEEP, удаление текущего токена x_i – правило $DELETE, добавление нового

Таблица 1. Пример генерации ошибок в предложении

Исходное предложение	Предложение с ошибками	Корректирующие правила
'Человеческий', **'дух'**, 'непостижимо', 'могуч', ',', 'и', 'убить', 'его', 'в', **'человеке'**, 'почти', 'невозможно', '.'	**'Человеческий-дух'**, 'непостижимо', 'могуч', ',', 'и', 'убить', 'его', 'в', **'чел', 'овеке'**, 'почти', 'невозможно', '.'	**$TRANSFORM_SPLIT_HYPHEN**, $KEEP, $KEEP, $KEEP, $KEEP, $KEEP, $KEEP, $KEEP, **$MERGE_SPACE**, $KEEP, $KEEP, $KEEP

Таблица 2. Информация об обучающем наборе данных

Набор данных	Число предложений	Доля предложений с ошибками	Этап обучения
Википедия + сочинения	10.000.000	≈ 100%	I
Лит. произведения + сочинения	1.000.000	≈ 50%	II

токена t_1 после текущего токена x_i – правило $APPEND\_t_1$ или замена текущего токена x_i на другой токен t_2 – правило $REPLACE\_t_2$. Для генерации правил $APPEND и $REPLACE использованы 2500 наиболее употребляемых русских слов с точки зрения коэффициента Жуайна [24]. Для каждого слова w_i добавлены соответствующие правила $APPEND_w_i и $REPLACE_w_i.

Грамматические правила выполняют операции, специфичные для конкретного случая. Для русского языка использованы следующие правила, которые применяются непосредственно к токену: изменение времени, падежа, рода, лица и числа. Добавлены правила для часто допускаемых ошибок, например, правописание “ться”/“тся” и “при”/“пре”. Также использованы правила, которые не зависят от языка: объединение двух слов с помощью пробела или дефиса; разделение слова, написанного через дефис, на две части; изменение регистра первой буквы слова.

Рассмотрим пример корректирующего правила для исправления “красивая” на “красивый”. Таким образом, необходимо преобразовать прилагательное из женского рода в мужской. Изначально правила для грамматических преобразований создавались следующим образом:

$TRANSFORM_ADJF_GEND_femn_masc

Такие правила содержали подробную информацию:

1) о части речи слова;
2) грамматическом признаке, который необходимо изменить;
3) граммеме как для исходного, так и для исправленного слова.

Далее было решено объединить правила для разных частей речи, исключив название части речи. Кроме того, решено отказаться от указания граммемы для исходного (неправильного) слова, поскольку для его исправления необходима информация только о новой граммеме. В данном исследовании предполагается, что модель сама научится сопоставлять правила соответственно частям речи. Например, глагол и прилагательное могут менять род, а существительное – нет. Таким образом, итоговые правила, используемые в нашей модели, выглядят следующим образом:

$TRANSFORM_GEND_masc

Данная модификация была сделана для того, чтобы уменьшить размер словаря и увеличить количество примеров для каждого правила в обучающем наборе данных. Таким образом достигается компромисс между обобщающей способностью и размером модели, так как с ростом количества правил увеличивается размер слоя классификации.

5. ЭКСПЕРИМЕНТЫ

В качестве предобученного кодировщика на основе архитектуры трансформер выбран Multilingual BERT. Обучение модели проводилось в течение двух этапов, каждый из которых длился 50 эпох. На первом этапе обучения размер батча равен 32, на втором – 16. В качестве оптимизатора параметров выбран Adam [25] с параметром $\mathrm{lr} = 10^{-5}$.

5.1. Примеры исправления предложений

В табл. 3 приведены примеры исправления ошибок моделью RuGECToR в предложениях, составленных человеком. Это результат применения соответствующих правил к входным токенам $\{x_1, x_2, \ldots, x_{n_i}\}$ в течение одной итерации. В результате работы модели слово “дораспределится” было исправлено в соответствии с правилами русского языка, хотя данное слово не было представлено в обучающем наборе данных.

5.2. Примеры итеративного исправления

В табл. 4 представлен пример работы итеративного исправления. Модель предсказывает правило $TRANSFORM_ПРЕ_ПРИ во время первой итерации и правило $APPEND_ли во время второй итерации. Хоть предложение уже после первой итерации является безошибочным, с частицей “ли” оно звучит лучше.

5.3. Примеры исправления реальных сочинений

В данном подразделе исследован результат работы модели RuGECToR на реальных сочинениях. Рассмотрены только те предложения,

Таблица 3. Примеры исправления предложений моделью RuGECToR

Корректирующее правило	Пример предложения
$KEEP	Я хочу играться
$TRANSFORM_ТЬСЯ/ТСЯ	Я хочу (игратся → играться)
$KEEP	Мы не успели дораспределиться
$TRANSFORM ТЬСЯ/ТСЯ	Мы не успели (дораспределится → дораспределиться)
$TRANSFORM_ПРЕ/ПРИ, $APPEND_ли	Можешь (ли) (приобразовать → преобразовать) это выражение?
$MERGE_SPACE	(Рас скажи → Расскажи) о себе
$TRANSFORM_GEND_femn	Она очень (красивый → красивая)
$MERGE_HYPHEN	(Красно зеленый → Красно-зеленый) цвет
$KEEP	Красный, зеленый цвета
$DELETE	Гектор (плохо → ∅) работает
$KEEP	Гектор отлично работает

Таблица 4. Пример итеративного исправления

#итерация	Исправление предложения	Позиция исправляемого токена
Исходное предложение	Можешь приобразовать это выражение?	0
Первая итерация	Можешь преобразовать это выражение?	2
Вторая итерация	Можешь ли преобразовать это выражение?	1

в которых модель обнаружила хотя бы одну ошибку. В табл. 5 приведены примеры таких предложений. Модель исправляет ошибки неидеально, но она обучалась на синтетически сгенерированных данных и тем не менее способна находить ошибки в реальных предложениях.

6. МЕТРИКИ КАЧЕСТВА

Сравнение качества работы моделей проводилось на синтетических и реальных данных. Для оценки качества использованы метрики, описанные в [11]:

$$R = \frac{\sum_{i=1}^{N} |g_i \cap e_i|}{\sum_{i=1}^{N} |g_i|}, \quad P = \frac{\sum_{i=1}^{N} |g_i \cap e_i|}{\sum_{i=1}^{N} |e_i|},$$

$$F_{0.5} = \frac{\left(1 + 0.5^2\right) \cdot R \cdot P}{R + 0.5^2 \cdot P},$$

где N – количество предложений; e_i – множество исправлений, предсказанных нашей моделью для

Таблица 5. Примеры исправления реальных сочинений

Исходное предложение	Исправленное предложение
Таким образом любовь к Родине крайне положительно влияет на человека дарит вдохновение в творчестве и **силы** в борьбе.	Таким образом любовь к Родине крайне положительно влияет на **жизнь** человека дарит вдохновение в творчестве и **сил** в борьбе.
В этом случае межнациональный конфликт подразумевает конфликт между этническими общностями, обычно проживающих поблизости государствах.	В этом случае межнациональный конфликт подразумевает конфликт между этническими общностями, обычно проживающих поблизости **в** государствах.
Автор повествует о бедной семье, в которой **пока** единственный ребенок – старший сын.	Автор повествует о бедной семье, в которой единственный ребенок – старший сын.
Этот случай показывает, что для родителей не может быть счастья большего, чем радость ребенка, а исполнение мечт и **вера** во что-то чудесное, не позволяет оксвернить несчастьям и ненастью светлую душу ребенка.	Этот случай показывает, что для родителей не может быть счастья большего, чем радость ребенка, а исполнение мечт и **веры** во что-то чудесное, не позволяет оксвернить несчастьям и ненастью светлую душу ребенка.

предложения s_i, а g_i – множество эталонных исправлений. Пересечение между g_i и e_i определяется как:

$$g_i \cap e_i = \{e \in e_i \mid \exists g \in g_i : e = g\}.$$

6.1. Синтетические данные

Для синтетического тестового набора данных сгенерировано 10 тыс. предложений с ошибками. Результаты на синтетическом наборе данных приведены в табл. 6. Видно, что метрики R и $F_{0.5}$ на втором этапе обучения ниже, чем на первом. Это можно объяснить двумя причинами:

1. 50% предложений в данных, использованных для второго этапа обучения, не содержали ошибок. Точность стала выше, но при этом количество охватываемых токенов уменьшилось.

2. На втором этапе обучения были использованы дополнительные предложения из литературных произведений, которые не использовались во время первого этапа обучения. Таким образом, была увеличена обобщающая способность модели за счет добавления данных из другого распределения.

6.2. Реальные данные

В качестве реальных тестовых данных использован набор данных RULEC. Результаты приведены в табл. 7. Модель достигает значения равного 22.2 с точки зрения метрики $F_{0.5}$. Это значение выше результата, демонстрируемого базовым подходом набора данных RULEC несмотря на то, что наша модель на нем не обучалась. Таким образом, модель обладает хорошей обобщающей способностью и не переобучается под конкретный набор данных. Следует также отметить, что на синтетических тестовых данных наша модель работает лучше, чем на реальных данных. Это вполне ожидаемо, так как синтетический обучающий и синтетический тестовый наборы данных имеют схожие распределения, в то время как RULEC сильно отличается от синтетического обучающего набора данных. Табл. 7 также показывает, что обобщающая способность модели на втором этапе обучения растет. На наборе данных RULEC качество работы модели после второго этапа значительно выше, чем после первого.

Таблица 6. Качество работы модели на синтетическом тестовом наборе данных

Модель	Этап обучения	P	R	$F_{0.5}$
RuGECToR	I	88.4	67.1	83.1
RuGECToR	II	88.5	65.1	82.5

Таблица 7. Сравнение качества работы моделей на наборе данных RULEC

Модель	Данные для обучения	P	R	$F_{0.5}$
Classifiers (learner)	RULEC	22.6	4.8	12.9
Classifiers (minimal sup.)	RULEC	38.0	7.5	21.0
MT	RULEC	30.6	2.9	10.6
RuGECToR	синтетические (I этап)	23.6	5.6	14.3
RuGECToR	синтетические (II этап)	**40.8**	**7.9**	**22.2**

7. ЗАКЛЮЧЕНИЕ

Проблема исправления грамматических ошибок хорошо изучена для английского языка, но недостаточно изучена для морфологически богатых языков из-за сложности исправления ошибок и малого количества размеченных данных. В данном исследовании представлена эффективная и интерпретируемая модель исправления грамматических ошибок для русского языка. Для этого составлен словарь с корректирующими правилами для русского языка и сгенерирован собственный синтетический набор данных, состоящий из предложений с ошибками. После этого изменен этап применения модели для использования этих правил. В качестве архитектуры модели взята наиболее эффективная Sequence Tagging модель для английского языка – GECToR. Наша модель достигла значений 82.5 на синтетических данных и 22.2 на реальных данных с точки зрения метрики $F_{0.5}$. Модель превосходит базовую модель для набора данных, на котором она не обучалась.

В дальнейшей работе мы хотим расширить словарь корректирующих правил и провести настройку параметров модели на других наборах данных. Также направлением дальнейших исследований является использование различных кодировщиков на основе архитектуры трансформер для русского языка и их ансамблирование.

СПИСОК ЛИТЕРАТУРЫ

1. *Rozovskaya A., Roth D.* Grammatical error correction: Machine translation and classifiers // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin, Germany: Association for Computational Linguistics, 2016. P. 2205–2215.

2. *Yuan Z., Stahlberg F., Rei M., Byrne B., Yannakoudakis H.* Neural and FST-based approaches to grammatical error correction // Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications. Florence, Italy: Association for Computational Linguistics, Aug. 2019. P. 228–239. [Online]. https://aclanthology.org/W19-4424
3. *Bryant C., Ng H.T.* How far are we from fully automatic high quality grammatical error correction? // ACL, 2015.
4. *Rajput D.* Review on recent developments in frequent itemset based document clustering, its research trends and applications // Int. J. Data Anal. Tech. Strateg. 2019. V. 11. P. 176–195.
5. *Flickinger D., Yu J.* Toward more precision in correction of grammatical errors // Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task. Sofia, Bulgaria: Association for Computational Linguistics, 2013. P. 68–73.
6. *Yuan X., Pham D., Davidson S., Yu Z.* ErAConD: Error annotated conversational dialog dataset for grammatical error correction // Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Seattle, United States: Association for Computational Linguistics, 2022. P. 76–84.
7. *Sie Yuen Lee J., Seneff S.* An analysis of grammatical errors in non-native speech in English // 2008 IEEE Spoken Language Technology Workshop, 2008. P. 89–92.
8. *Zhuravlev K., Rudakov K., Inyakin A., Kirsanov A., Lisitsa A., Nikitov G., Peskov N., Yaminov R., Chekhovich Y.* The system of recognition of intellectual text reuse "antiplagiat" // Mathematical methods of pattern recognition: 12th All-Russian conference: Collection of reports. MAKS Press, 2005. P. 329–332.
9. *Keck C.M.* How do university students attempt to avoid plagiarism? A grammatical analysis of undergraduate paraphrasing strategies // Writing & Pedagogy. 2010. V. 2. P. 193–222.
10. *Zhang W.E., Sheng Q.Z., Alhazmi A., Li C.* Adversarial attacks on deep learning models in natural language processing: A survey // ACM Trans. Intell. Syst. Technol., 2020. V. 11. № 3.
11. *Ng H.T., Wu S.M., Briscoe T., Hadiwinoto C., Susanto R.H., Bryant C.* The CoNLL-2014 shared task on grammatical error correction // Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014. P. 1–14. [Online]. https://aclanthology.org/W14–1701.
12. *Rozovskaya A., Roth D.* Grammar error correction in morphologically rich languages: The case of Russian // Transactions of the Association for Computational Linguistics. 2019. V. 7. P. 1–17.
13. *Rothe S., Mallinson J., Malmi E., Krause S., Severyn A.* A simple recipe for multilingual grammatical error correction // Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers). Online: Association for Computational Linguistics. Aug. 2021. P. 702–707.
14. *Grundkiewicz R., Junczys-Dowmunt M., Heafield K.* Neural grammatical error correction systems with unsupervised pre-training on synthetic data // Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications. Florence, Italy: Association for Computational Linguistics, 2019. P. 252–263.
15. *Omelianchuk K., Atrasevych V., Chernodub A., Skurzhanskyi O.* GECToR – grammatical error correction: Tag, not rewrite // Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications. Seattle, WA, USA. Online: Association for Computational Linguistics, 2020. P. 163–170.
16. *Malmi E., Krause S., Rothe S., Mirylenka D., Severyn A.* Encode, tag, realize: High-precision text editing // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, 2019. P. 5054–5065.
17. *Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L., Polosukhin I.* Attention is all you need // Advances in Neural Information Processing Systems, I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., 2017. V. 30. Curran Associates, Inc.
18. *Korobov M.* Morphological analyzer and generator for russian and ukrainian languages // Analysis of Images, Social Networks and Texts, ser. Communications in Computer and Information Science, M. Khachay, N. Konstantinova, A. Panchenko, D. Ignatov, and V. Labunets, Eds. Springer International Publishing. 2015. V. 542. P. 320–332.
19. *Devlin J., Chang M.-W., Lee K., Toutanova K.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Volume 1 (Long and Short Papers), 2019. P. 4171–4186. Minneapolis, Minnesota. Association for Computational Linguistics.
20. *Vrandečić D., Krötzsch M.* Wikidata: A free collaborative knowledgebase // Communications of the ACM. Sep 2014. V. 57. № 10. P. 78–85.
21. Open source collection of school essays. https://www.kritika24.ru, accessed: 07.11.2022.
22. Open source collection of literary works. https://proza.ru, accessed: 07.11.2022.

23. *Trinh V.A., Rozovskaya A.* New dataset and strong baselines for the grammatical error correction of Russian // Findings of the Association for Computational Linguistics: ACL–IJCNLP 2021. Online: Association for Computational Linguistics, 2021. P. 4103–4111.
24. *Lyashevskaya O., Sharov S.* Frequency Dictionary of the Modern Russian Language (based on the materials of the National Corpus of the Russian Language) [in Russian]. M.: Azbukovnik, 2009.
25. *Kingma D.P., Ba J.* Adam: A method for stochastic optimization, 2017.

RUGECTOR: RULE-BASED NEURAL NETWORK MODEL FOR RUSSIAN LANGUAGE GRAMMATICAL ERROR CORRECTION

© 2024 I. A. Khabutdinov[a, b], A. V. Chashchin[b], A. V. Grabovoy[a, b], A. S. Kildyakov[b], Y. V. Chekhovich[b, c]

[a] *Moscow Institute of Physics and Technology (National Research University) Institutskiy per. 9, Dolgoprudny, Moscow Region, 141701 Russia*

[b] *Antiplagiat Company Varshavskoe highway 33, Moscow, 117105 Russia*

[c] *Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Vavilova st. 44-2, Moscow, 119333 Russia.*

Grammatical Error Correction is one of the core Natural Language Processing tasks. At the moment, the open source state-of-the-art Sequence Tagger for English is GECToR model. For the Russian language, this problem does not have solutions with the same good results due to the lack of labeled datasets, therefore we decided to contribute to the aforementioned task. In this research, we described the process of creating a synthetic dataset and training a model on it. We adapt GECToR architecture for Russian language and call it RuGECToR. We use this architecture because, unlike Sequence-to-Sequence approach, it is easy to interpret and does not require a lot of training data. Our goal was to train the model in such a way that it does not adapt to a specific sample, but generalizes the morphological properties of the language. The presented model achieves an $F_{0.5}$ of 82.5 on synthetic data and an $F_{0.5}$ of 22.2 on RULEC dataset which was not in the training set.

Keywords: natural language processing, grammatical error correction, sequence tagging, datasets creation, fine-tuning