Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 5637–5647
Marseille, 20-25 June 2022

Automatic Classification of Russian Learner Errors

Alla Rozovskaya
Queens College
City University of New York
arozovskaya@qc.cuny.edu

Abstract

Grammatical Error Correction systems are typically evaluated overall, without taking into consideration performance on individual error types because system output is not annotated with respect to error type. We introduce a tool that automatically classifies errors in Russian learner texts. The tool takes an edit pair consisting of the original token(s) and the corresponding replacement and provides a grammatical error category. Manual evaluation of the output reveals that in more than 93% of cases the error categories are judged as correct or acceptable. We apply the tool to carry out a fine-grained evaluation on the performance of two error correction systems for Russian.

Keywords: grammatical error correction, Russian, error classification

1. Introduction

In the field of Grammatical Error Correction (GEC), similarly to Machine Translation, evaluation is performed by first producing *edits* (token deletions, insertions, and replacements) needed to transform the source sentence into its corrected counterpart. This is done by aligning the two sentences. A *gold edit* is an edit between a source sentence and its reference, i.e. a corrected version produced by a human annotator. A *proposed (system) edit* is an edit between a source sentence and the corrected version proposed by the automatic system, or system hypothesis. A *correct edit* is an edit in the intersection of gold and proposed edits.

Below we show a sample (source, reference) pair, a possible alignment, and the resulting *gold edits*: "reallistic" $\rightarrow$ "realistic", "a" $\rightarrow$ $\varnothing$), and "are" $\rightarrow$ "were" .

- Source: The settings ***are*** very ***reallistic*** and the actors had ***a*** great performance .
- Ref.: The settings ***were*** very ***realistic*** and the actors had great performance .

When annotating and correcting learner data, annotators can also be asked to categorize the resulting gold edits. For the example above, the edits would be classified as verb tense, spelling, and missing determiner.

Classifying system edits generated by neural machine translation frameworks that are being used today is not trivial, as these systems are not restricted in the type of edits that can be made (Susanto et al., 2014; Yuan and Briscoe, 2016; Hoang et al., 2016; **?**; Junczys-Dowmunt and Grundkiewicz, 2016; Mizumoto and Matsumoto, 2016; Jianshu et al., 2017; Chollampatt and Ng, 2018).

As the system edits are not labeled for linguistic type, it is not possible to perform type-based evaluation. Type-based evaluation would be extremely useful, as it could help provide directions for making further progress in system development by allowing one to focus on and improve performance on specific error types. Furthermore, the resulting error categories can be used in language learning applications, to provide educational feedback. Finally, automatic type classification allows for a standartization of multiple GEC datasets (Bryant et al., 2017). For instance, various GEC corpora in English (e.g. FCE (Yannakoudakis et al., 2011) and CoNLL (Ng et al., 2014)) have different error classification schemas, which can be unified using automatic error typing.

Recently, Bryant et al. (2017) introduced ERRANT, an error annotation toolkit for English, designed to extract edits from a pair of the original and corrected sentences and classify these according to a rule-based framework. Grammatical error types are assigned based on rules that rely on the part-of-speech (POS) of original and corrected token(s), as opposed to manually designed error categories. A key advantage of a rule-based approach over a canonical automatic one of training a classifier is that this approach is not tied to a specific error classification schema, which can vary significantly among datasets in the same language (Bryant et al., 2017). ERRANT is widely used for fine-grained evaluation and error analysis in English GEC (Bryant et al., 2019).

Our approach is inspired by ERRANT and adapted to the specific challenges of a language with rich morphology, such as Russian. We develop a rule-based classification framework, which uses POS and morphological information to classify edits. Manual evaluation of the resulting edits with 2 raters and on 2 learner corpora shows that the edits are judged as correct or acceptable on average 93.5% of the time. We use the tool to perform fine-grained evaluation of two GEC systems using two learner corpora. Type-based evaluation allows us to identify performance differences between the two models, as well as to determine the sets of "easy" and "challenging" errors for the cur-

Corpus	**Total words**	**Incorr. words**	**Error rate (%)**
RULEC (Foreign)	164,071	11,343	6.9
RULEC (Herit.)	42,187	1,705	4.0
RU-Lang8 (dev)	23,138	3,605	15.6
RU-Lang8 (test)	31,603	3,558	11.3

Table 1: Error rates in RULEC (foreign and heritage speakers shown separately) and in RU-Lang8. *Error rates* refer to the percentage of tokens that have been modified.

rent systems. We show, in particular, that most of the learner errors, with the exception of spelling mistakes and select morpho-syntactic errors, remain challenging. Overall, errors that require major changes that go beyond character-level modifications remain unresolved. The tool will be made available for research.

2. Automatic Error Typing

2.1. Learner datasets

We perform experiments using two datasets of Russian learner data manually corrected for errors: the RULEC-GEC corpus (Rozovskaya and Roth, 2019) (henceforth RULEC) and another dataset of Russian learner writing that has been recently collected from the online language learning platform Lang-8 (Mizumoto et al., 2011) and annotated by native speakers. We refer to this dataset as RU-Lang8 (Trinh and Rozovskaya, 2021).

RU-Lang8 is collected from the Lang-8 website and contains data from learners of a variety of foreign languages. The size of the Russian subcorpus is 633,000 tokens. A subset of that (54,000 tokens) has been manually corrected by expert annotators and split into development and test. The RU-Lang8 corpus differs from RULEC: the latter consists of essays written on a University setting in a controlled environment, while the Lang8 data was collected online; the majority of texts are short paragraphs or questions posed by learners.

To compare to RULEC, we show in Table 1 the error rates, i.e. the percentage of the erroneous tokens in the data. RU-Lang8 data has significantly higher error rates than both foreign and heritage parts of RULEC (Table 1). We attribute this to the overall higher proficiency level of the RULEC corpus writers.

2.2. Automatic edit extraction

Before the edits can be categorized, they need to be extracted using pairs of original and corrected sentences. This is essentially an alignment problem, where the start and end position of each edit need to be identified. Table 2 shows a pair of original and corrected sentence and a sample alignment that includes three edits: (1) merging the first two tokens всё таки $\rightarrow$ всё-таки, (2) merging tokens 4 and 5 не плохо $\rightarrow$ неплохо and (3) inserting missing preposition $\emptyset$ $\rightarrow$ на. We note that this

Всё **таки**	он	не плохо	играет		скрипке
Всё-таки	он	**неплохо**	играет	**на**	скрипке
Nevertheless	he	not badly	plays	on	the violin

Table 2: Sample alignment between original and corrected sentence. The source and the target token(s) in each edit are shown in red and blue, respectively.

Dataset	**Method**	**Edit Extraction**		
		P	**R**	F_1
RULEC	Lev.	88.0	85.9	87.0
	DL-Rules	91.0	88.1	**89.5**
RU-Lang8	Lev.	82.8	80.1	81.4
	DL-Rules	91.4	86.1	88.7

Table 3: Performance of different edit extraction methods. Lev stands for Levenshtein distance; DL-Rules stands for Damerau-Levenshtein distance and linguistically-motivated rules.

is not a unique alignment. For example, inserting the missing preposition can be represented as скрипке $\rightarrow$ на скрипке.

In the humanly-corrected data, edit boundaries are typically already known, however, for automatically-corrected data this is not the case. Thus, automatic edit extraction is necessary to identify boundaries of the proposed edits.[1]

Several attempts have been made at automatic edit extraction. Swanson and Yamangil (2012) used Levenshtein distance, but it does not align multi-token edits, i.e. edits where there is more than one token on either side. Felice et al. (2016) proposed a rule-based linguistically-motivated alignment algorithm. It includes both generic and English-specific rules combined with the Damerau-Levenshtein (DL) algorithm and showed the effectiveness of the approach compared to Levenshtein distance.[2]

We evaluate the rule-based strategy (excluding rules that are English-specific) and the Levenshtein distance. We compare the quality of the automatically-extracted edits against the gold humanly-generated edits in Table 3. Surprisingly, the Levenshtein distance performs reasonably well on the Russian datasets, obtaining F-scores of 81-87% on the RU-Lang8 and RULEC datasets, respectively (on the English datasets it achieved and an F-score of 60%). The rule-based strategy is better on RULEC by 2 points and is a 7-point improvement on RU-Lang8.

Overall, results on edit extraction on Russian are slightly better than on English. This may be due to the fact that single-token edits account for a higher percentage of Russian edits (91.4% in RULEC and 93.8%

[1] Automatic alignment may also desirable for humanly-corrected data to ensure uniformity across learner datasets (Felice et al., 2016)

[2] DL algorithm, unlike Levenshtein, is able to handle word order (transposition) errors.

(1) Determine the original token(s) (O) and the corrected token(s) (C). Check whether O and C are in the dictionary.
(2) If O is not in the dictionary, assign category *Spelling*.
(3) Deletions and Insertions: A single-token insertion or deletion is categorized based on the token's POS. Exceptions are nouns, verbs, and adjectives that are assigned the category Insert or Delete. Insertion or deletion of a punctuation symbol („-?!) is treated as Punctuation error.
(4) Replacements
(a) Replacing one punctuation symbol with another is Punctuation category.
(b) If C is not in the dictionary, the error category is based on POS tag of the token, e.g. NO/AO/VO (Noun Other). Note that other POS categories are closed-class and thus words that are not in the dictionary can only be a noun, verb, or adjective/adverb.
(c) Determine POS tags of the O and C tokens. If one of the tokens has more than one POS tags select those that match. If POS tags do not match, the error category is either Lexical or Morphology (see below).
(d) If POS categories match but base words are different, error category is Lexical or Morphology (see below) for N, V, and A. All other words are assigned error type based on POS tag.
(e) Distinguish between Morphology and Lexical errors: different POS tags or different base forms for A, N, or V. Compute the length of the longest common prefix (or suffiex) in O and C. If its greater than or equal to the half of the O or C (whichever is less), the error type is Morphology. The intuition here is the stem word is most likely the same, and the error involves incorrect affixation (e.g.).
(f) For O and C that share the same POS tag and base form, error type is assigned based on the mismatch between linguistic categories within the specific POS tag. For example, if both O and C are nouns, the values of the categories gender, case, and number are compared. If the values for number are different, for instance, the error category assigned is NN. Multiple categories are assigned if more than one category does not match.
(g) Verb aspect errors: In addition to category mismatch as in (f), the aspectual value of the verb is determined based on a list of verb that lists all pairs of corresponding verbs (perfective/imperfective). This is needed since the morphological analysis maps verbs that share the same stem but different aspectual forms to different base forms. Using the aspect list allows us to identify pairs of verbs that, even though they do not share the same base form, they differ only in the aspect category and thus we classify these as Aspect errors.
(h) Verb voice errors: These are difficult to identify as these are not mapped to the same base form. We thus check for the specific endings that signal the passive (reflexive) verbs (собрать ("get something ready")$\rightarrow$ собрать-ся ("get oneself ready"). The ending ся signals reflexive form.

Table 4: Error classification rules. O stands for original token, and C stands for corrected token.

in RU-Lang8), while in English, these account for 71-81%. A lot of multi-token edits in English are phrasal verbs, verbs with auxiliaries, and missing possessive markers on nouns. These categories are typically expressed in Russian via morphological markers. In the remainder of the paper, we use the DL-rules method of edit extraction.

2.3. Automatic type prediction

Error type classification in English relies on the POS information of the edit tokens (Bryant et al., 2017). Since Russian is a morphologically-rich language, we rely on the grammatical categories within each major POS (V(erb), N(oun), A(djective)). This approach follows closely the error categories adopted for the manual annotation in Rozovskaya and Roth (2019).

Overview of the rules The complete description of rules is presented in Table 4. The pseudocode is presented in Appendix Algorithm 1. We adhere to the morphological properties within each major POS category (A, N, V); otherwise the POS tag is used to assign error type (preposition, pronoun, adverb, etc.). The original and corrected tokens are pre-processed with a morphological analyzer (Sorokin, 2017). Given a word token, the analyzer produces the token's base form and generates all possible morphological analyses. Each morphological analysis consists of a concatenation of all the relevant grammatical categories. For nouns and adjectives, the categories are *gender* (masculine/feminine/neuter), *number* (singular/plural), and *declension* (6 cases). For verbs, the categories include *aspect* (perfective/imperfective), *voice* (active/reflexive), *number* (singular/plural), *gender* (masculine/feminine/neuter), *person* (1st, 2nd, 3rd), *tense* (past/present/future), *voice* (active/reflexive), and may include additional categories, such as gerund or participle.[3] Many Russian words do not have a unique analysis (we discuss this further below).

Given the list of grammatical properties for the source

[3]Participles behave like adjectives in Russian and are specified for the same set of categories. We treat them as adjectives.

Error type	RULEC		RU-Lang8	
	Gold count	Rel. Freq.	Gold count	Rel. freq.
Spelling	965	18.3	739	21.8
Lex. choice	819	15.5	497	14.8
Punc.	592	11.2	277	8.2
Prep.	333	6.3	236	7.0
Replace	381	7.2	185	5.5
Insert	252	4.8	144	4.3
Delete	94	1.8	117	3.5
Adv.	78	1.5	90	2.7
Conj.	77	1.5	78	2.3
Part.	56	1.1	36	1.1
Morph.	110	2.1	30	0.9
Noun case	339	6.4	217	6.4
Noun case/Noun number	423	8.1	144	4.3
Noun number	29	0.6	14	0.4
Noun other	13	0.3	-	-
Noun (all)	804	15.2	382	11.3
Verb aspect	50	1.0	82	2.4
Verb other	68	1.3	75	2.2
Verb agreement	134	2.5	73	2.2
Verb tense	12	0.2	16	0.5
Verb voice	45	0.9	12	0.4
Verb (all)	324	6.1	264	7.8
Adj. case	101	1.9	62	1.8
Adj. gender	69	1.3	70	2.1
Adj. other	128	2.4	83	2.5
Adj. number	55	1.0	49	1.4
Adj. (all)	370	7.0	275	8.1

Table 5: List of automatic error types and their distribution in the RULEC and RU-Lang8 corpora.

and the target tokens, the error category is determined based on the property that has different values in the source and target tokens. For example, given an edit that involves two nouns that share the same base form, a noun case error is assigned if the number and gender properties match, but the case values do not.

Several error types require additional checks – morphology, lexical choice, and spelling. If a source token is not in the dictionary[4] we consider the edit to be a spelling mistake. Insertions and deletions that involve open-class words (N, V, A) are marked as Insert and Delete, respectively. Otherwise, an inserted or deleted token is assigned the error type based on its POS tag. *Lexical change* errors involve single-token edits that do not share a stem, while *Replace* errors are mistakes that involve multi-token replacements.

Challenges specific to Russian: Morphology errors Word morphology errors include edits where the source and the target token share the same stem but incorrect word formation. Some of the mistakes in this category result in words that are not valid Russian words (due to an incorrect word formation suffix), and, since these are not found in the dictionary, can be confused with spelling mistakes (путешеств-овая $\rightarrow$ путешеств-уя, "while travelling" where an incorrect suffix is used resulting in an invalid word.). We view such errors separately from spelling errors. Some morphology errors involve confusing valid words that share the same stem but differ in morphology (e.g. практич-еский "practic", практич-ный "practical").[5] Identifying morphology errors is difficult overall since the morphological analyzer does not map such edits to the same base form. We use an approach of identifying the longest common subsequence between the source and the target, which allows us to catch some of these errors. Nevertheless, distinguishing between morphological errors and spelling errors is challenging when the source token is not a valid word.

Challenges specific to Russian: Noun number and case errors As mentioned above, some Russian surface forms have multiple analyses. A notable case is nouns: for some nouns, the singular genitive form, the plural nominative, and the plural accusative forms are the

[4]We use a large native corpus of Russian (250M tokens collected over the web (Borisov and Galinskaya, 2014) to build a dictionary of valid Russian words.

[5]Morphology errors also occur in English, but in Russian, arguably, these are more common and challenging due to more options of morphological formation.

Rater	RULEC			RU-Lang8		
	Good	Accept.	Bad	Good	Accept.	Bad
1	70	25	5	88	8	4
2	63	25	12	83	12	5

Table 6: Manual evaluation of the automatically assigned error categories by each rater and each dataset on a set of 100 edits, randomly selected.

Model	RULEC			RU-Lang8		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
CNN	55.8	26.6	45.7	57.9	26.8	**47.0**
Transf.	63.3	27.5	**50.2**	55.3	28.5	46.5

Table 7: Results on the test of the models trained on learner and synthetic data. Best result for each test set is in bold.

same.[6] This ambiguity can be resolved in most cases by considering sentential context, however, we only look at token edits in isolation. Such cases are problematic, as, depending on the analysis chosen, a different mismatch in the grammatical category between the source and the target tokens will be identified (either number or case). For such edits, we predict both noun case and noun number error categories. A manual evaluation of 31 ambiguous cases shows that 64.5% of these are errors in case, while 22.6% are errors in noun number, and 12.9% have errors in both number and case.

Manual evaluation of automatic error typing We perform a manual evaluation of the rule-based classifier, following the approach in Bryant et al. (2017) used to evaluate the performance of ERRANT. 100 randomly chosen edits from gold reference files are manually evaluated by two independent native annotators as “Good”, “Acceptable”, or “Bad”. “Good” means the chosen error type was the most appropriate for the given edit; “Acceptable” means that the chosen type was appropriate but not the optimum, and “Bad” means not appropriate.
The results of the evaluation are shown in Table 6. In all cases, with the exception of RU-Lang8 and rater 2, at least 95% of the predicted error types were judged as acceptable. These results are comparable to those reported for English. However, note that, unlike with ERRANT, our evaluation excludes trivial error categories, such as Punctuation, Insertion, and Deletion errors. A complete list of the error categories and statistics on the two learner datasets are shown in Table 5.

3. Type-Based Evaluation

We now apply the automatic error classification to the outputs of two GEC models applied two learner corpora: RULEC and RU-Lang8. The models that make use of state-of-the-art techniques: a Convolutional Encoder-Decoder Neural Network (*CNN*) (Chollampatt and Ng, 2018) and a Transformer model (Naplava and Straka, 2019). We train the CNN model using the same hyperparameter values. The Transformer achieved the highest F-score on the RULEC corpus in the literature.
Both of the models use the training and dev portions of RULEC as learner data, and similar amounts of native data with synthetic errors. Overall performance of the two models on each of the two benchmark corpora is shown in Table 7. Observe that the transformer model outperforms CNN on RULEC by almost 5 points, however, on the RU-Lang8 corpus the two models perform similarly.
We show type-based performance of the models on RULEC and RU-Lang8 in Tables 8 and 9, respectively. For both datasets, highest results are achieved on spelling mistakes (F-scores between 60% and 70%). This is followed by errors related to grammar on nouns, verbs, and adjectives (F-scores of 40-60%). The Transformer model also does well on preposition errors, while the CNN model performs well on punctuation mistakes.
Regarding performance differences between the models, the Transformer does better on RULEC in almost all error categories, with the exception of punctuation. However, on RU-Lang8, both models perform similarly on errors related to nouns, verbs, and adjectives, with the CNN model being slightly ahead, which suggests that the Transformer may be overfitting towards RULEC more strongly than the CNN model. Finally, the most challenging error categories for both models are lexical choice, replacement,[7] and insertion errors. To summarize, the type-based analysis indicates that the models currently have difficulty correcting mistakes that involve major changes that go beyond character-level modifications.

4. Conclusion

We describe a tool for automatic classification of learner errors in Russian. The classification is based on POS (similar to English) but also incorporates additional constraints specific to Russian. We believe this tool can be easily modified to fit another (morphologically-rich) language. The tool has been evaluated on two Russian learner datasets with encouraging results. Using the tool, we have also conducted type-based evaluation of two state-of-the-art GEC systems. Type-based evaluation can assist in making progress on developing robust GEC systems. In particular, we showed that beyond spelling mistakes and select grammar errors that require character-level modifications, current systems do not do well on major learner misuse that requires more global lexical changes. This tool can also be used to inform language learners and provide feedback with explanations.

[6]This is one common ambiguity but is not the only one.

[7]Replacements refer to multi-token lexical changes.

Error type	CNN			Transformer		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Spelling	66.2	53.9	63.3	75.93	63.73	**73.13**
Lex. choice	46.3	3.0	12.1	67.07	13.43	**37.29**
Punc.	54.8	23.3	**43.1**	42.71	6.93	21.01
Replace	0.00	0.00	0.00	2.30	1.05	1.86
Prep.	25.2	8.1	17.7	70.25	25.53	**52.02**
Morph.	20.0	1.8	6.7	51.61	14.55	**34.19**
Insert	0.0	0.0	0.0	17.39	6.35	**12.90**
Delete	75.0	3.2	13.6	38.24	13.83	**28.26**
Noun (all)	61.1	38.4	54.6	72.0	36.4	**60.2**
Verb (all)	54.5	20.4	40.8	71.5	38.0	**60.8**
Adj (all)	50.0	21.6	39.6	64.1	29.5	**51.9**

Table 8: Evaluation by error type on the RULEC corpus.

Error type	CNN			Transformer		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Spelling	66.9	45.8	61.3	70.3	63.3	**68.8**
Lex. choice	38.8	3.8	13.7	50.00	13.68	**32.7**
Punc.	67.1	39.0	**58.6**	44.4	5.8	19.0
Prep.	49.2	14.0	32.7	76.2	20.3	**49.2**
Replace	0.0	0.0	0.0	9.6	10.3	**9.7**
Morph.	0.0	0.0	0.0	50.0	16.7	**35.7**
Insert	25.0	2.1	7.8	12.2	6.2	**10.2**
Delete	76.5	11.1	**35.1**	39.2	17.1	31.1
Noun (all)	69.6	41.9	**61.4**	70.5	35.1	58.7
Verb (all)	54.8	19.3	40.1	49.1	21.2	38.9
Adj. (all)	74.8	33.5	**60.0**	66.4	34.5	56.1

Table 9: Evaluation by error type on the RU-Lang8 corpus.

5. Bibliographical References

Borisov, A. and Galinskaya, I. (2014). Yandex school of data analysis russian-english machine translation system for WMT14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics.

Bryant, C., Felice, M., and Briscoe, T. (2017). Automatic annotation and evaluation of error types for grammatical error correction. In *ACL*.

Bryant, C., Felice, M., Andersen, Ø., and Briscoe, T. (2019). The BEA-19 shared task on grammatical error correction. In *Proceedings of the ACL Workshop on Innovative Use of NLP for Building Educational Applications (BEA-19)*.

Chollampatt, S. and Ng, H. T. (2018). A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the AAAI*. Association for the Advancement of Artificial Intelligence.

Chollampatt, S., Taghipour, K., and Ng, H. T. (2016). Neural network translation models for grammatical error correction. In *IJCAI*.

Felice, M., Bryant, C., and Briscoe, T. (2016). Automatic extraction of learner errors in esl sentences using linguistically enhanced alignments . In *COLING*.

Hoang, D.-T., Chollampatt, S., and Ng, H.-T. (2016). Exploiting n-best hypotheses to improve an SMT approach to grammatical error correction. In *IJCAI*.

Jianshu, J., Wang, Q., Toutanova, K., Gong, Y., Truong, S., and J. Gao, J. (2017). A nested attention neural hybrid model for grammatical error correction. In *ACL*.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2016). Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *EMNLP*.

Mizumoto, T. and Matsumoto, Y. (2016). Discriminative reranking for grammatical error correction with statistical machine translation. In *NAACL*.

Mizumoto, T., Komachi, M., Nagata, M., and Matsumoto, Y. (2011). Mining revision log of language learning SNS for automated japanese error correction of second language learners. In *IJCNLP*.

Naplava, J. and Straka, M. (2019). Grammatical error correction in low-resource scenarios. In *W-NUT Workshop*.

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., and Bryant, C. (2014). The conll-2014 shared task on grammatical error correction. In *Pro-*

ceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pages 1–14, Baltimore, Maryland, June. Association for Computational Linguistics.

Rozovskaya, A. and Roth, D. (2019). Grammar error correction in morphologically-rich languages: The case of russian. In *Transactions of ACL*.

Sorokin, A. (2017). Spelling correction for morphologically rich language: a case study of russian. In *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*.

Susanto, R. H., Phandi, P., and Ng, H. T. (2014). System combination for grammatical error correction. In *EMNLP*.

Swanson, B. and Yamangil, E. (2012). Correction detection and error type selection as an esl educational aid. In *NAACL-HLT*.

Trinh, V. A. and Rozovskaya, A. (2021). New dataset and strong baselines for the grammatical error correction of russian. In *ACL Findings*.

Yannakoudakis, H., Briscoe, T., and Medlock, B. (2011). A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA, June. Association for Computational Linguistics.

Yuan, Z. and Briscoe, T. (2016). Grammatical error correction using neural machine translation. In *NAACL*.

Appendix

Algorithm 1

Error Type Classification Algorithm

```
Input: Source token(s) $s$, target token(s) $t$, baseform dictionary $BaseD$ containing all basic wordforms and their POS tags as keys, and a list of all possible morphological analyses and corresponding wordforms; wordform dictionary $WordD$ containing a morphological form as key and a list of possible analyses with corresponding POS tag as values; list of verb aspect pairs AspectD listing perfective/imperfective verb pairs; list of punctuation symbols PUNC
Output: error type //Set list of grammatical categories
gram_cats=['case','number','gender','tense','aspect','voice','form','finite']
//Handle split and merge errors
if $t$ equals $s$.replace(' ','') or $s$ equals $t$.replace(' ','') then
    return Spelling
end if
//If source or target contain more than one token, set error to Replace
if len($s$.split())>1 or len($t$.split())>1 then
    return Replace
end if
//Punctuation errors
if $s$ in $PUNC$ or $t$ in $PUNC$ then
    return Punctuation
end if
//Spelling errors if source word not in dictionary
if $s$ not in WordD then
    return Spelling
end if
//Missing token errors
if $s$ is empty then
    Initialize POS(target) $\leftarrow$ None
    $POS(target) = WordD[t]$ if $t$ in WordD
    if $POS(target)$ in {Adj,Noun,Verb} then
        return Insert
    else
        return POS(target)
    end if
end if
//Extraneous token errors
if $t$ is empty then
    Initialize POS(source) $\leftarrow$ None
    $POS(source) = WordD[s]$ if $s$ in WordD
    if $POS(source)$ in {Adj,Noun,Verb} then
        return Delete
    else
        return POS(source)
    end if
end if
//Get list of all baseforms and their corresponding POS tags for source token
Initialize pos_list(source) $\leftarrow$ None
Initialize baseform_list(source) $\leftarrow$ None
$pos\_list(source) = WordD[s].postags()$
$baseform\_list(source) = WordD[s].baseforms()$
//Handle errors where target word not in dictionary
if $t$ not in WordD then
    $POS(source) = WordD[s]$
    if $POS(source)$ in {Adj,Noun,Verb} then
        return POS(source)$+' : O'$
    else
        return POS(source)
    end if
end if
//Get list of all baseforms and their corresponding POS tags for target token
Initialize pos_list(target) $\leftarrow$ None
Initialize baseform_list(target) $\leftarrow$ None
$pos\_list(target) = WordD[t].postags()$
$baseform\_list(target) = WordD[t].baseforms()$
//Find common POS of source and target tokens
Initialize pos $\leftarrow$ None
Initialize baseform_t $\leftarrow$ None
Initialize baseform_s $\leftarrow$ None
for pos pos_t in pos_list(target) do
    if pos_t in pos_list(source) then
        pos=pos_t //Set baseforms of source and target tokens based on the common POS
        base(s)=WDict[s].getbase(pos)
        base(t)=WDict[t].getbase(pos)
    end if
end for
//If source and target do not have the same POS, error is either M or L
//Check if s and t share a common prefix or suffix that is at least half of min(len(s),len(t))+1
if pos equals None then
    $Lcom = 1 + \frac{min(len(s),len(t)}{2}$
    if Lcom$\geq$ 4 and (s[: $Lcom$] equals t[: $Lcom$] or s[$-Lcom$ :] equals t[$-Lcom$ :] ) then
        return M
    else
        return L
    end if
end if
//If shared POS is not an Adj., Noun, or Verb
if pos not in {A,N,V} then
    return pos
end if
```

Algorithm 1

Error Type Classification Algorithm: Part II

```
Input: Source token(s) $s$, target token(s) $t$, baseform dictionary $BaseD$ containing all basic wordforms and their POS tags as keys, and a list of all possible morphological analyses and corresponding wordforms; wordform dictionary $WordD$ containing a morphological form as key and a list of possible analyses with corresponding POS tag as values; list of verb aspect pairs AspectD listing perfective/imperfective verb pairs; list of punctuation symbols PUNC
Output: error type
//Handle verb voice errors – confusing active and reflexive forms
if pos equals V then
    for ending en in {ся,сь} do
        if base(s)+en equals base(t) or base(s) equals base(t)+en then
            return VV
        end if
    end for
end if
//Handle verb aspect errors using verb aspect table of perfective/imperfective verb pairs
if pos equals V then
    if base(s) in DAspect and base(t) in DAspect[base(s)] then
        return VA
    end if
end if
//Source and target have same POS, different baseforms – check if it could be an M error //Check if s and t share a common prefix or suffix that is at least half of min(len(s),len(t))+1
if base(s) not equals base (t) then
    $Lcom = 1 + \frac{min(len(s),len(t))}{2}$
    if Lcom$\geq$ 4 and (s[: $Lcom$] equals t[: $Lcom$] or s[$-Lcom$ :] equals t[$-Lcom$ :] ) then
        return M
    else
        return L
    end if
end if
//Get all morph. variants of baseform with chosen POS; note that here source and target share the same baseform and POS
morpho_variants=BaseD[pos+':'+baseform(s)]
cat_values={}
for gr in gram_cats do
    cat_values[gr]={}
    cat_values[gr][0]=cat_values[gr][2]=''
    cat_values[gr][1]=cat_values[gr][3]=0
end for
for var in morpho_variants do
    word=var.word()
    if word in {s,t} then
        gram_cats_word=var.values()
        //Example category is GENDER; example value is FEM
        for c in gram_cats_word do
            c_name=gram_cats_word.name()
            c_val=gram_cats_word.value()
            if c_name in gram_cats then
                index=0
                if word equals t then
                    index=2
                end if
                //Add parameter value for the corresponding (source or target) wordform
                if c_val not in cat_values[c_name][index]  then
                    cat_values[c_name][index]+=':'+c_val
                    cat_values[c_name][index+1]+=1
                end if
                if pos equals 'V' and c_val equals 'NFIN' then
                    inf+=1
                end if
            end if
        end for
    end if
end for
//Handle errors where verb infinitival form is confused with tensed form
if inf equals 1 then
    return VINF
end if
```

Algorithm 1

Error Type Classification Algorithm: Part III

```
Input: Source token(s) $s$, target token(s) $t$, baseform dictionary $BaseD$ containing all basic wordforms and their POS tags as keys, and a list of all possible morphological analyses and corresponding wordforms; wordform dictionary $WordD$ containing a morphological form as key and a list of possible analyses with corresponding POS tag as values; list of verb aspect pairs AspectD listing perfective/imperfective verb pairs; list of punctuation symbols PUNC
Output: error type
//Compare values of grammatical categories for source and target, where POS is either adjective, noun, or verb; and set error category based on the category where the values in source and target differ
cats_N = {'case':'NC','num':'NN','gen':'NG'}
cats_V = {'aspect':'VA','tense':'VT','num':'VNG','gen':'VNG','per':'VNG','voice':'VV','other':'VO','case':'VO'}
cats_A = {'case':'AC','num':'AN','gen':'AG','form':'AO'}
error_type=''
for gr in cat_values do
    if (pos equals N and gr in cats_N) or (pos equals A and gr in cats_A) or (pos equals V and gr in cats_V) then
        if cats_values[gr][1]*cats_values[gr][3] greater than 0 then
            if pos equals N then
                diff_val=cats_Ngr
            else if pos equals A then
                diff_val=cats_Agr
            else if pos equals V then
                diff_val=cats_Vgr
            end if
            if diff_val not in error_type then
                error_type+=diff_val+','
            end if
            values=cat_values[gr][0].split(':') //Check if source and target wordforms share same parameter value
            for val in values do
                if (val not equal '') and (val in cat_values[gr][2]) then
                    cat_values[gr][1]=0
                    Break
                end if
            end for
            //If source and target wordforms do not share the value for given parameter, set error type to this parameter
            if cat_values[gr][1] not equal 0 then
                if pos equals N then
                    diff_val=cats_Ngr
                else if pos equals A then
                    diff_val=cats_Agr
                else if pos equals V then
                    diff_val=cats_Vgr
                end if
                if diff_val not in error_type then
                    error_type+=diff_val+','
                end if
            end if
        end if
    end if
end for
if error_type equals '' then
    error_type=pos+':O'
end if
return error_type
```