Language-specific Correction Engine

1st Vlad-Cristian Matei
Computer Science Department
University Politehnica of Bucharest
Bucharest, Romania
vlad_cristian.matei@stud.acs.upb.ro

2nd Mircea Timpuriu
Computer Science Department
University Politehnica of Bucharest
Bucharest, Romania
mircea.timpuriu@stud.acs.upb.ro

I. Introduction

To meet the requirements of milestone 2, we aim to present a portion of the essential literature that serves as inspiration for developing our own experiments within the task of Grammatical Error Correction (GEC). We will start by discussing the main target dataset and its alternatives, the types of errors, and the methods used to obtain a parallel dataset, consisting of synthetically altered versions and the correct variants. Finally, we will focus on the training process using an architecture that performs Neural Machine Translation (NMT).

II. Datasets

A. MARCELL

The MARCELL corpus [1], more specifically the MARCELL-RO Romanian text subcorpus represents the body of recent Romanian national legislation collected up to the year of 2021. The main focus of this dataset was collecting Romanian governmental decisions, ministerial orders, decisions, decrees, and laws. Even though this corpus can range back to laws passed in the 19th century, we aim to collect datasets that are grammatically correct in the world of today, and as a result, we will only take into consideration the section of this corpus that incorporates legislation passed in the last 30 years.

Furthermore, beyond the grammatical correctness of this dataset, a major factor considered when choosing this dataset relates to the very well-organized structure of the dataset. The MARCELL-RO subcorpus has been split into sentences, each sentence being thoroughly analysed, and each word being annotated with information that is very relevant for our cause, such as the part of speech, the lemma, case, gender, number, form etc. These annotations will prove to be extremely important, especially within the detection module (GED) of the Grammatical Error Corrector, because the grammatical part-of-speech analysis is a crucial first step for detecting possible errors at word level.

B. Romanian Written Language Corpus

The Romanian Language Written Corpus [2] is a monitor corpus with an approximate size of 5 million words, containing a wide range of Romanian literary works, from all domains and genres, ranging from belles-lettres to historical, scientific, and general interest writing, from novels and poems and fiction etc. The domain of literary works is another domain, other than law, in which we trust the datasets to be grammatically correct, but also, one more crucial aspect has been taken into account while choosing this dataset.

By using this dataset, we also aim to introduce the concept of diversity within the Romanian Language. The country of Romania consists of multiple regions, each with their own defining aspects when it comes to the Romanian Language spoken in that region, so it is crucial that we include written samples from all types of registers of the Romanian Language. The Romanian Language Written Corpus incorporates 16 writing registers, and works from around 380 different authors, mentioned in the metadata of the dataset files.

C. Europarl

Europarl [3] is a parallel corpus extracted from the proceedings of the European Parliament. It includes versions in 21 European languages: Romance (French, Italian, Spanish, Portuguese, Romanian), Germanic (English, Dutch, German, Danish, Swedish), Slavic (Bulgarian, Czech, Polish, Slovak, Slovene), Finno-Ugric (Finnish, Hungarian, Estonian), Baltic (Latvian, Lithuanian), and Greek.

The purpose of extracting and processing this corpus was to generate sentence-aligned texts for statistical machine translation systems. The dataset contains a significant amount of text in Romanian: approximately 400k sentences with around 10 million words. Additionally, there is also a parallel corpus with translations between English and Romanian.

D. RoNAC

The authors [1] introduce, to the best of their knowledge, the first corpus for Romanian Grammatical Error Correction (GEC). The Romanian National Audiovisual Council Corpus (RONACC) is presented as a native corpus, encompassing corrected sentences spoken on Romanian TV and radio shows, alongside additional sentence pairs representing common written mistakes.

The corpus is categorized into three groups: NAC-phrases, NAC-sentences, and W-sentences. NAC-phrases consist of phrases lacking verbs, NAC-sentences comprise well-formed

[1] https://marcell-project.eu/

[2] https://github.com/lmidriganciochina/romaniancorpus

[3] https://www.statmt.org/europarl/

sentences from spoken language, and W-sentences include well-formed sentences with written mistakes. Each phrase in the corpus is designed to contain sufficient information for correction without the need for additional context. The total number of GOLD labeled sentences is approximately 10k.

III. Types of Errors

Primarily, we've opted to approach the GEC task as an NMT problem. Consequently, the dataset will consist of parallel data, featuring correct sentences on one side and their corresponding counterparts with grammatical errors on the other. Due to the challenge of manually annotating tens of thousands of sentences, we will attempt to generate our dataset artificially. We will start with correct sentences and endeavor to automatically introduce various errors. The challenge at hand is figuring out how to programmatically generate grammatical errors that closely resemble those made by humans.

A. ERRANT

The taxonomy presented in ERRANT [2] suggests that the majority of error types are related to part-of-speech (POS) categories. As such, approximately 50 rules are defined for generating grammatical errors based on automatic POS tags. These categories cover a wide range of common grammatical errors and provide a detailed framework for their classification and correction.

ERRANT type	Example
MORPH	în cazul unei paciente [**internată** → internate] joi
ORTH	Acum, în [**camera deputaților** → Camera Deputaților]
SPELL	treceți la ceea ce [**vroiați** → voiați] să ziceți
ORDER	Nu [**mai o** → o mai] da cotită
OTHER	permis de [**port-armă** → portarmă]
POS:NOUN	O roțiță care să-și aducă [**aportul** → contribuția].
POS:NOUN:FORM	să vedem cum a fost din [**punct** → punctul] de vedere al organizării [...]
POS:VERB	opoziția ar putea [**demara** → începe] procedura de suspendare
POS:VERB:FORM	Omul negru poate [**fii** → fi] folosit metaforic [...]
POS:ADJ	Mulți se tem astăzi să se întâlnească cu personaje dubioase sau într-un context [**aiurea** → nepotrivit].
POS:ADJ:FORM	E un pic [**ambiguu** → ambiguă] definirea termenului mită aici
POS:ADV	Și te costă și foarte, foarte [**ieftin** → puțin]
POS:PRON	o să trebuiască să dați niște mesaje, [_ → niște] telefoane [...]
POS:PRON:FORM	Pe [**aceiași** → aceeași] bandă circulai?
POS:DET	șeful [_ → unui] aerodrom privat din Comana
POS:DET:FORM	de liderii PSD, PNL și [**a** → ai] minorităților naționale
POS:ADP	80% [**din** → dintre] victimele traficanților
POS:CCONJ	dânsa ca [**și** → _] persoană luptătoare
POS:PUNCT	lovește mingea înapoi[_ → ,]dar au început să se apropie

Fig. 1. ERRANT Error Categories for Romanian Text [1]

B. The Taxonomy of the Four Categories

Presented at the *Ukrainian Natural Language Processing Workshop* [3], the possible errors that can be generated are now categorized into four high-level categories: **punctuation**, **spelling**, **grammar**, and **fluency**. Especially within the grammar errors category, we can find the classification from the ERRANT taxonomy. However, we find that certain error categories, such as fluency, punctuation, or spelling, deserve somewhat separate treatment, hence finding this suitable taxonomy.

Error type	Description
Grammar-related errors	
G/Case	incorrect usage of case of any notional part of speech
G/Gender	incorrect usage of gender of any notional part of speech
G/Number	incorrect usage of number of any notional part of speech
G/Aspect	incorrect usage of verb aspect
G/Tense	incorrect usage of verb tense
G/VerbVoice	incorrect usage of verb voice
G/PartVoice	incorrect usage of participle voice
G/VerbAForm	incorrect usage of an analytical verb form
G/Prep	incorrect preposition usage
G/Participle	incorrect usage of participles
G/UngrammaticalStructure	digression from syntactic norms
G/Comparison	incorrect formation of comparison degrees of adj. and adverbs
G/Conjunction	incorrect usage of conjunctions
G/Other	other grammatical errors
Fluency-related errors	
F/Style	style errors
F/Calque	word-for-word translation from other languages
F/Collocation	unnatural collocations
F/PoorFlow	unnatural sentence flow
F/Repetition	repetition of words
F/Other	other fluency errors

Fig. 2. Examples of grammar and fluency errors. [3]

Having this taxonomy makes it easier to consider what percentage of errors should be generated from each category for a more realistic distribution.

Error type	Total	%
Grammar (all)	6,682	14.4
Fluency (all)	10,924	23.6
Spelling	8,771	19.0
Punctuation	19,871	43.0

Fig. 3. The Density of Error Types for the Ukrainian Language. [3]

Thus, in order to better understand the Romanian language, we attempted to draw inspiration from Ukrainian or Russian. Russian is a language that has influenced the Romanian language over the years, and it is also characterized by rich morphology and a large number of inflections and conjugational forms of parts of speech.

In [4], the authors enlisted two annotators, native speakers of Russian with a background in linguistics, to correct a dataset called RULEC (12k sentences). At the end, they presented the error distribution for native Russian speakers and those who speak Russian as a second language, and compared it with the statistics of two popular English datasets: FCE and CoNLL. The results are shown in Fig 4. Additional information about the W&I+LOCNESS dataset can be found in [5].

Top errors (%)		
Russian	**English (FCE)**	**English (CoNLL-test)**
Spell. 21.7	Art. 11.0	Lex. choice 14.2/14.4
Noun:case 13.2	Lex. choice 9.5	Art. 13.9/13.3
Lex. choice 12.3	Prep. 9.0	Mechan. 9.6/14.9
Punc. 9.6	Spell. 8.1	Noun:number 9.0/6.8
Miss. word 8.4	Punc. 8.0	Prep. 8.8/11.7

Fig. 4. Comparison statistics for Russian and English learner corpora [4]

Additionally, comparing native speakers to non-native speakers helps us understand the influence each category of individuals has, enabling us to generate synthetic data specific to the target category we are focusing on.

Foreign			Heritage		
Error	(%)	Errors per 1,000	Error	(%)	Errors per 1,000
Spell.	18.6	11.7	Spell.	42.4	15.7
Noun:case	14.0	8.8	Punc.	22.9	8.5
Lex. choice	13.3	8.3	Noun:case	7.8	2.9
Miss. word	8.9	5.6	Lex. choice	5.5	2.0
Punc.	7.6	4.8	Miss. word	4.7	1.7
Replace	6.3	3.9	Replace	2.8	1.0
Extra word	5.7	3.5	Extra word	2.4	0.9
Adj:case	3.9	2.4	Adj:case	2.1	0.8
Prep.	3.3	2.1	Word form	2.1	0.8
Word form	3.1	2.0	Noun:number	1.8	0.7
Noun:number	2.6	1.6	Verb agr.	1.6	0.6
Verb agr.	2.5	1.6	Prep.	1.5	0.6

Fig. 5. Most common errors for foreign and heritage Russian speakers [4]

IV. Methods for Generating Synthetic Data

There are many ways to alter grammatically correct sentences. The survey [6] views this automatic perturbation as both a general method of **Noise injection**, achieved, for example, through **Rule-based alterations**, and a way to obtain erroneous sentences through methods such as **Back-translation** or **Round-trip Translation**.

A. Rule-based alterations

Using this method typically covers most errors in the grammar and spelling categories. A very interesting approach is presented in the article [7]. The authors of the study present a probability distribution of error generation based on the length of the sentence.

Length	Err.	Prob.	Length	Err.	Prob.
$[1,3)$	0	0.50	$[6,9)$	2	0.30
	1	0.50		3	0.45
$[3,6)$	1	0.50		4	0.25
	2	0.50	$[16,20)$	3	0.10
$[9,16)$	3	0.15		4	0.15
	4	0.25		5	0.15
	5	0.30		6	0.30
	6	0.30		7	0.30
$[20,30)$	4	0.10	$[30,\infty)$	5	0.10
	5	0.15		6	0.15
	6	0.15		7	0.15
	7	0.30		8	0.30
	8	0.30		9	0.30

Fig. 6. Probability distribution of sentence errors [7]

Then, after determining the number of errors to be generated, the type of error to be generated is also established using a probability distribution, in a hierarchy as presented in the figures below:

1) **Concatenation**: combine two consecutive tokens.
2) **Misspelling**: introduce spelling errors into words.
3) **Substitution**: seven different types of substitutions are presented, including substitution between prepositions, articles, singular pronouns, plural pronouns, etc.
4) **Deletion**: delete the token.
5) **Transposition**: the token exchange position with a consecutive token.

Additionally, when it comes to **misspelling**, a variety of errors can be used, with some of the most common categories being *deletion*, *insertion*, *transposition*, and *replacement*.

Type	Prob.
Concatenation	0.12
Misspell	0.45
Substitution	0.40
Deletion	0.00
Transposition	0.03

Fig. 7. Token Error types [7]

Tok. length	Err.	Prob.
$[1,3)$	0	1.00
$[3,5)$	1	1.00
$[5,10)$	1	0.80
	2	0.20
$[10,\infty)$	1	0.75
	2	0.15
	3	0.10

Fig. 8. Number of misspells in a token [7]

Type	Prob.
Deletion	0.30
Insertion	0.15
Transposition	0.25
Replacement	0.30

Fig. 9. Mispell types [7]

Another method of error generation, based on rules, is presented in the article [8]. Synthetic errors are introduced into an error-free text as follows: For each sentence, an error probability is sampled from a normal distribution with a mean of 0.15 and a standard deviation of 0.2. This probability, multiplied by the sentence length, determines the number of words to change. For each selected word, one of four operations (substitution, deletion, insertion, or swapping with an adjacent word) is performed with specified probabilities. Word substitution has a probability of 0.7, while the other operations have a probability of 0.1 each.

index		","	;	:	—	-	.	?	!	…
	0.95	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
","	0.41	0.59	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
;	0.80	0.09	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00
:	0.86	0.05	0.00	0.05	0.04	0.00	0.00	0.00	0.00	0.00
—	0.87	0.05	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00
-	0.80	0.00	0.00	0.00	0.03	0.17	0.00	0.00	0.00	0.00
.	0.16	0.00	0.00	0.00	0.00	0.00	0.84	0.00	0.00	0.00
?	0.80	0.00	0.00	0.00	0.00	0.00	0.04	0.12	0.04	0.00
!	0.80	0.00	0.00	0.00	0.00	0.00	0.10	0.10	0.00	0.00
…	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20

Fig. 10. Punctuation error probability matrix [9]

To synthetically generate **punctuation errors**, the authors of [9] create an error probability matrix. This matrix replaces each punctuation mark (including spaces between words) with any other punctuation mark according to a randomly generated probability. Subsequently, this matrix is applied to

each sentence in the dataset. The resulting matrix reflects the probabilities of replacing each punctuation mark with another one.

Similarly, the authors of [9] seek to refine a rule that ensures the **grammatical accuracy** of Ukrainian sentences. This rule stipulates that adjectives, verbs, or pronouns within a sentence must align in gender, number, and case with the noun, abbreviation, or pronoun that acts as the head of the phrase. **Regarding fluency disruption**, the authors have devised two rules: *replacing written numbers with their symbols* and *word inversions*.

Error Type	Cost	Error Type	Cost
Substitution – horizontally adjacent letters	2.290	Substitution – diacritic redundancy	2.250
Substitution – vertically adjacent letters	2.661	Substitution – other cases	4.285
Substitution – $z \rightarrow s$	2.747	Insertion – horizontally adjacent letter	2.290
Substitution – $s \rightarrow z$	1.854	Insertion – vertically adjacent letter	2.661
Substitution – $y \rightarrow i$	3.167	Insertion – same letter as previous	1.227
Substitution – $i \rightarrow y$	2.679	Insertion – other cases	2.975
Substitution – non-adjacent vocals	3.706	Deletion	4.140
Substitution – diacritic omission	2.235	Swap letters	3.278

Fig. 11. Korektor spelling specific errors [10]

Therefore, all these rules along with their hyperparameters need to be used with specificity for the Romanian language and adapted to the chosen dataset. An interesting example of adaptability is found in the construction of the Czech spell-checker Korektor [10], which calculates a cost for several very specific spelling errors well adapted to the language or the arrangement of letters on the keyboard.

B. Back-translation

The authors of [11] highlight the scarcity of resources for the Ukrainian language, a challenge shared with Romanian. Nevertheless, they propose that employing the back-translation method offers a solution by enabling the generation of a diverse array of errors, almost exclusive to the language. This approach has the potential to focus on rectifying fluency-related errors. In their study, the back-translation process begins with sentences sourced from a native Ukrainian corpus. These sentences undergo translation into another language (pivot) before being translated back into Ukrainian. English serves as the pivot language, generating top n translation hypotheses for each sentence. Subsequently, top m back-translations into Ukrainian are produced for each hypothesis. Each unique Ukrainian word is then matched with the back-translated words, which are considered as potential synthetic errors.

Bridge Language	CoNLL-2014			JFLEG
	Precision	Recall	$F_{0.5}$	GLEU$^+$
Round-Trip Translations				
French	33.6	21.9	30.3	50.6
German	36.4	21.2	31.8	51.3
Russian	33.5	21.1	30.0	50.5
Japanese	35.7	51.3	38.1	46.2
All	38.1	27.1	35.2	52.1

Fig. 12. Round-trip translation experiment [12]

Another approach to back-translation is presented in [13], where the authors explore an alternative method of generating alterations. In this method, the noisy model is trained using the inverse of the GEC parallel dataset, where ungrammatical sentences are treated as the target and grammatical ones as the source.

Also, in the article [12], back-translation is presented as a round-trip translation where multiple languages are experimented with, exploring various scenarios and drawing different conclusions, thereby understanding different correlations. In the example below, a significant influence on errors is observed from translations from Japanese for the CoNLL dataset.

C. Round-translation

The final technique explored in [9] was round translation, known for its effectiveness with low-resource monolingual datasets. This process involved tokenizing and translating a sentence from Ukrainian to Russian using the Marian UK-RU encoder and transformer. The translated sentence was then back-translated using the UK-RU (instead of RU-UK) tokenizer and the correct transformer. This approach results in errorful sentences generated with an incorrect tokenizer, producing sentences resembling a mix of Russian and Ukrainian. It is important to investigate whether this method can also benefit the Romanian language.

D. Combining different systems

Titled *"Learning to combine Grammatical Error Corrections"*, the article [14] presents various approaches to using different grammar correction systems. Combining these systems iteratively has led to significantly improved results, as depicted in the figure below.

System	P	R	$F_{0.5}$
(1) Nematus1	0.4788	0.1544	0.3371
(2) Nematus2	0.4839	0.1583	0.3429
(3) Nematus3	0.4842	0.1489	0.3338
(4) Nematus4	0.4843	0.1502	0.3352
(5) Spellchecker	0.5154	0.0308	0.1242
(6) Bert	0.0132	0.0147	0.0135
1+2	0.4972	0.1854	0.3721
1+2+3	**0.5095**	0.1904	0.3816
1+2+3+4	0.4926	0.2017	0.3824
1+2+3+4+5	0.5039	0.2233	0.4027
1+2+3+4+5+6	0.5029	**0.2278**	**0.4051**

Fig. 13. Iterative combination of systems [14]

V. Architectural Models

During this section, we are going to look at possible architectures for either or both correction and detection modules of our grammatical correction engine, those being the **Transformer architecture**, the **Copy-augmented architecture** and the **Sequence tagging method**, and ultimately express our opinions and take some decisions regarding the best solutions for our own correction engine.

A. *Transformer architecture*

The first method we are going to look at in terms of implementing the correction engine is the simple Transformer architecture. The Transformer stands as the base of many new developments regarding the GEC task, and the Natural Language Processing field as a whole, but it also remains relevant on its own, as a standalone architecture, to this very day.

As we can see in Fig. 14, it is an auto-regressive model [15], composed by an encoder that maps an input sequence to continuous representations in a vector space, and a decoder that generates an output sequence one element at a time based on the encoder data, using the concept of self-attention, an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence.

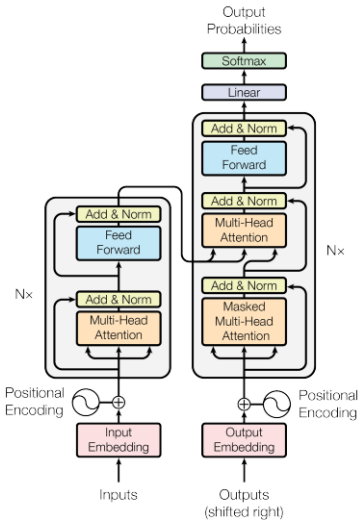

Fig. 14. Transformer architecture [15]

This sort of basic Transformer model has been implemented in article [1], where it was developed forward in order to extract some of the best results for the Romanian Grammatical Error Correction task. As a first baseline test, a 1.5 million parameter Transformer was trained on the data extracted in that article. Then, for gauging the performance of the model, a larger Transformer model has been initially pre-trained on an artificially generated dataset, the differences between the two models being seen in Fig. 15.

Parameters	Transformer-tiny	Transformer-base
Encoder layers	3	6
Decoder layers	3	6
Embeddings size	128	512
Filter size	128	2048
Attention heads	2	8
Dictionary size	2k	32k
Dropout	0.2	0.1

Fig. 15. Transformer model characteristics [1]

Then, to further maximize the performance, one of two ways is chosen: either the optimizer and the learning rate is reset before further training on a gold corpus, or these settings are left untouched before the further training.

These different performance enhancing methods, along with the different decoding strategies in the article, bring out the best a simple Transformer model has to offer and also bring a sort of diversity to the training process that is very much needed for a successful architecture.

B. *Copy-augmented architecture*

Another architecture taken into account is a model also based on a Transformer, namely the Copy-Augmented Architecture, first presented in the article [16]. The article suggests that, when a person attempts to correct a sentence, it doesn't go out of its way to process each and every word of a sentence, and just aims to correct the words it considers to be wrong, and merely copying the rest of the words, the correct ones. Furthermore, the data presented in Fig. 16 shows us that in some of the most used datasets, and generally in most languages, the differences between correct sentences and generally flawed sentences are very small, the word similarity percentage between two parallel sentences in these two categories and, implicitly, the percentage of words that would need to be simply copied and not modified between sentences is quite high.

Corpus	Sent.	Tok.	Same %
CoNLL-2013	1,381	28,944	96.50%
JFELG	754	14,240	84.23%
Lang-8	4,936	73,705	83.22%

Fig. 16. Similarity percentage between parallel sentence corpora [16]

Thus, the aim has been to implement this mentality into a Grammatical Error Correction module. So, as described in Fig. 17, besides implementing an attention-based Transformer that uses the context of a sentence to choose its next words from a probability distribution over the whole vocabulary of the model, it also has the ability to copy words from the input sequence, thus improving the efficiency of the model by reducing the need to generate the whole sequence from scratch.

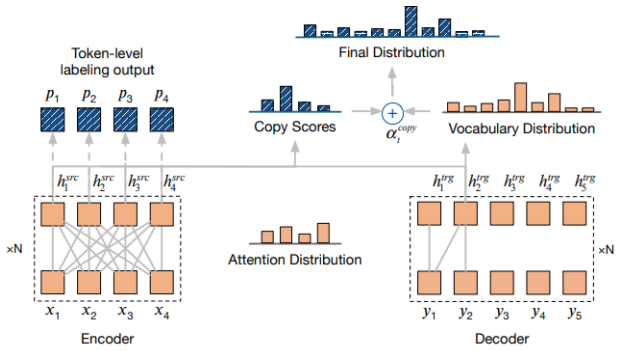

Fig. 17. Copy-augmented architecture [16]

This mentality is by no means a new concept, being first implemented for the text summarization task inside the Pointer-Generator Network architecture presented in article [17], but it has also found its logical use for the Grammatical Error Correction task, because of the sheer nature of a task that requires text correction, as we mentioned earlier.

The main mechanism of a Copy-Augmented Architecture is that, at every word generation step, the model has the choice to either generate a new word from the vocabulary based on the context, or simply copy a word from the input sequence, each action being executed with a specific probability that takes into consideration multiple factors related to the nature of the model and of the data used. The combination between this probability and the probability distributions employed by both choices will eventually determine the next word in the sentence.

C. Sequence tagging

Lastly, we are considering a Sequence Tagging model, specifically for the Grammatical Error Detection (GED) module of our correction engine. Having the correct text and the synthetic text generated from it as the input, we can align the words of a target sentence over its associated input sentence and detect the discrepancies between them and tag the erroneous sequences accordingly.

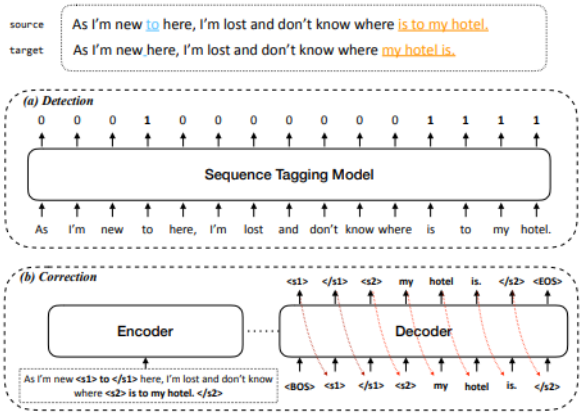

Fig. 18. Sequence tagging for GED [18]

The article [18] uses this Sequence Tagging architecture during the Erroneous Span Detection section, as visualised in Fig. 18, stating that this model solves the efficiency bottleneck of past architectures by feeding only the erroneous sequences to the correction module of the engine, represented in the aforementioned paper by a simple seq2seq neural translation model, rather than having it process the sentences whole, thus importantly reducing the processing time during training and overall improving performances for this task.

VI. Conclusions

First off, we have brought into discussion a wide variety of datasets fit for the needs of our engine. Considering the size, and also the amount of work that has been put into carefully and thoroughly annotating this dataset, we believe that the MARCELL-RO legislative corpus will represent the best dataset for training our models, a corpus we will regard as a core dataset for our work. Eventually, we will complement it with the rest of the aforementioned datasets, namely the Written Language Corpus, Europarl and RoNAC.

Our decision is to prioritize the taxonomy of the four categories due to its alignment with the structure of the Romanian language and its effectiveness in controlling the probability distribution of errors. This taxonomy allows us to prioritize the most common errors, such as punctuation, fluency, and spelling, which are critical for our correction engine.

While rule-based methods effectively cover grammar, spelling, and punctuation errors, back-translation offers a solution for generating a diverse array of language-specific errors, particularly enhancing fluency-related corrections. However, our belief is that the most effective results will emerge from well-constructed datasets employing complex, efficient, and diverse error taxonomies and generation techniques.

Given this, we conclude that a simple yet highly relevant architecture, like a base Transformer model, is sufficient for our correction engine's needs. It not only meets our requirements but also allows us to compare our work with other research efforts focused on creating new datasets. Additionally, we acknowledge the potential of the copy-augmented architecture as an alternative, provided we have the resources to explore it in detail.

References

[1] T.-M. Cotet, S. Ruseti, and M. Dascalu, "Neural grammatical error correction for romanian," 11 2020, pp. 625–631.

[2] C. Bryant, M. Felice, and T. Briscoe, "Automatic annotation and evaluation of error types for grammatical error correction," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, R. Barzilay and M.-Y. Kan, Eds. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 793–805. [Online]. Available: https://aclanthology.org/P17-1074

[3] O. Syvokon, O. Nahorna, P. Kuchmiichuk, and N. Osidach, "UA-GEC: Grammatical error correction and fluency corpus for the Ukrainian language," in *Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP)*, M. Romanyshyn, Ed. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 96–102. [Online]. Available: https://aclanthology.org/2023.unlp-1.12

[4] A. Rozovskaya and D. Roth, "Grammar error correction in morphologically rich languages: The case of Russian," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 1–17, 2019. [Online]. Available: https://aclanthology.org/Q19-1001

[5] M. White and A. Rozovskaya, "A comparative study of synthetic data generation methods for grammatical error correction," in *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, J. Burstein, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, H. Yannakoudakis, and T. Zesch, Eds. Seattle, WA, USA → Online: Association for Computational Linguistics, Jul. 2020, pp. 198–208. [Online]. Available: https://aclanthology.org/2020.bea-1.21

[6] C. Bryant, Z. Yuan, M. R. Qorib, H. Cao, H. T. Ng, and T. Briscoe, "Grammatical error correction: A survey of the state of the art," *Computational Linguistics*, p. 1–59, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1162/coli_a_00478

[7] S. Xu, J. Zhang, J. Chen, and L. Qin, "Erroneous data generation for grammatical error correction," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational*

Applications, H. Yannakoudakis, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, and T. Zesch, Eds. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 149–158. [Online]. Available: https://aclanthology.org/W19-4415

[8] R. Grundkiewicz, M. Junczys-Dowmunt, and K. Heafield, "Neural grammatical error correction systems with unsupervised pre-training on synthetic data," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, H. Yannakoudakis, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, and T. Zesch, Eds. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 252–263. [Online]. Available: https://aclanthology.org/W19-4427

[9] M. Bondarenko, A. Yushko, A. Shportko, and A. Fedorych, "Comparative study of models trained on synthetic data for Ukrainian grammatical error correction," in *Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP)*, M. Romanyshyn, Ed. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 103–113. [Online]. Available: https://aclanthology.org/2023.unlp-1.13

[10] M. Richter, P. Straňák, and A. Rosen, "Korektor – a system for contextual spell-checking and diacritics completion," in *Proceedings of COLING 2012: Posters*, M. Kay and C. Boitet, Eds. Mumbai, India: The COLING 2012 Organizing Committee, Dec. 2012, pp. 1019–1028. [Online]. Available: https://aclanthology.org/C12-2099

[11] F. Palma Gomez, A. Rozovskaya, and D. Roth, "A low-resource approach to the grammatical error correction of Ukrainian," in *Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP)*, M. Romanyshyn, Ed. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 114–120. [Online]. Available: https://aclanthology.org/2023.unlp-1.14

[12] J. Lichtarge, C. Alberti, S. Kumar, N. Shazeer, N. Parmar, and S. Tong, "Corpora generation for grammatical error correction," in *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)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 3291–3301. [Online]. Available: https://aclanthology.org/N19-1333

[13] J. Ye, Y. Li, Y. Li, and H.-T. Zheng, "MixEdit: Revisiting data augmentation and beyond for grammatical error correction," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 10 161–10 175. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.681

[14] Y. Kantor, Y. Katz, L. Choshen, E. Cohen-Karlik, N. Liberman, A. Toledo, A. Menczel, and N. Slonim, "Learning to combine grammatical error corrections," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, H. Yannakoudakis, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, and T. Zesch, Eds. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 139–148. [Online]. Available: https://aclanthology.org/W19-4414

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[16] W. Zhao, L. Wang, K. Shen, R. Jia, and J. Liu, "Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data," in *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)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 156–165. [Online]. Available: https://aclanthology.org/N19-1014

[17] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *CoRR*, vol. abs/1704.04368, 2017. [Online]. Available: http://arxiv.org/abs/1704.04368

[18] M. Chen, T. Ge, X. Zhang, F. Wei, and M. Zhou, "Improving the efficiency of grammatical error correction with erroneous span detection and correction," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 7162–7169. [Online]. Available: https://aclanthology.org/2020.emnlp-main.581