UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE DEPARTMENT

RESEARCH REPORT II

Generating Erroneous Text for Natural Language Corrections

Teodor-Mihai Coteț

Thesis advisor:

Prof. dr. ing. Mihai Dascălu

BUCHAREST

2019

CUPRINS

Abstract 2

1 Introduction 3

2 State of the Art 4

2.1 Wikipedia edits 4

2.2 Artificially induced errors 5

2.3 Inducing noise with neural methods 5

3 Method 6

3.1 Wikipedia edits 6

3.2 Artificially induced errors 7

3.3 Inducing noise with neural methods 13

4 Results for Detecting and Correcting Word Level Errors 14

4.1.1 Models for correcting word level mistakes 15

4.1.2 Models for detecting word level mistakes 16

5 Future Work 19

Bibliography 20

ABSTRACT

This research report is focused on exploring methods to generate erroneous text which imitates as much as possible human written mistakes. The language in which the mistakes are generated is Romanian, with few equivalent experiments for English. This work is motivated by a lack of any annotated corpus with natural language mistakes for Romanian, but also for many other languages. Three main methods are explored: generating artificial errors using language-specific tags, filtering Wikipedia edits that contain grammatical errors, and generating artificial errors by modifying neural generative models, such as encoder-decoder. The methods used to generate these datasets can be classified into two categories: a) methods requiring a small starting human-annotated dataset, and b) methods requiring no previous dataset. The best models achieve over 97% accuracy in detecting mistakes, coupled with a 74% accuracy for the automated correction model.

1 INTRODUCTION

Grammatical error correction represents the task of automatically detecting and correcting grammatical errors of all kinds. Grammatical errors consist of many different types, including articles or determiners, prepositions, noun form, verb form, subject-verb agreement, pronouns, word choice, sentence structure, punctuation, capitalization, etc.

Although annotated corpora for grammatical errors exists for English, for other languages the situation is not as fortunate. For Romanian there is no publicly available such corpus. The main reason for this lack of resources is the difficulty of making such a corpus. Such a corpus requires a human expert which has to not only identify a mistake, but also come up with a correction, or even more.

Although error corrections systems can be designed without an annotated dataset, recent research (Xie, Avati, Arivazhagan, Jurafsky, & Ng, 2016) is focused on using neural networks as the main model for this task, thus requiring a large annoated samples to train the network, achieving near human performance (Grundkiewicz & Junczys-Dowmunt, 2018).

All word presented in the following sections uses the *MaxMatch* (M^2) scorer (Dahlmeier & Ng, 2012), which is an algorithm that computes the sequence of edits between source sentence and target sentence that achieves the highest overlap with the gold standard annotation. This scoring method has the main advantage that it takes into account the gold standard edits when searching for the system edits.

Methods based on Wikipedia edits and neural generation of artificial errors require a small human-annotated corpus (approx. 20k samples). Although no such corpus is present on this moment in Romanian, this work assumes that such a corpus will exists in the near future.

The general pipeline of the current and the future work is presented in Figure 1.

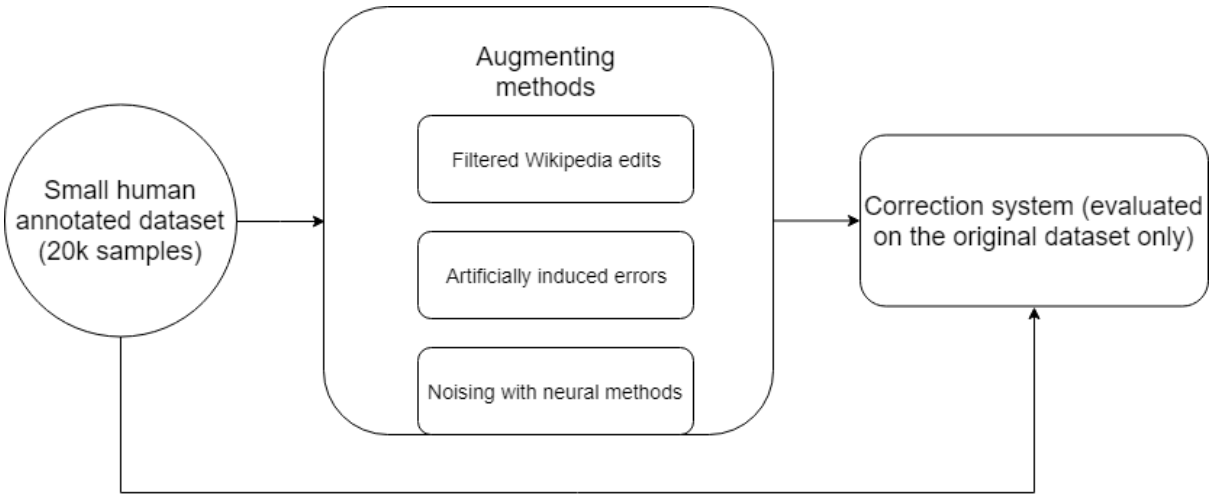

Figure 1: General pipeline.

2 STATE OF THE ART

Although there are annotated datasets for this specific problem in English, more is always better, especially when using neural networks. Thus, there is helpful research that explores ways in which an annotated dataset can be extended. Unless mentioned otherwise, the work presented below is performed for English.

2.1 Wikipedia edits

The first method of such extension is using Wikipedia edits history as a starting set with potential grammatical errors (Boyd, 2018), which are filtered and classified based on simple heuristics that take into account an original small (approx. 20k examples) dataset. The rules used for filtering are: This method has the benefit that Wikipedia edits are a free resource and are numerous for any language. In fact, the experiments in the previously mentioned paper were conducted using German for which the author mentions there is no publicly available annotated dataset for such a task. On the one hand, in the experiments in the paper show prove that even adding unfiltered Wikipedia edits improve de score $F_{0.5}$ with *4.13,* for 250k edits. However, for 500k unfiltered edits decreases slightly, by 0.8 the $F_{0.5}$ compared to using 250k edits. On the other, using 500k filtered Wikipedia edits improves the $F_{0.5}$ by *1.78*, compared to using 500k unfiltered edits. This shows that although using unfiltered Wikipedia improves the performance, filtering them will further improve this performance. The main advantage of this method is that it introduces in the corpora natural occurring error by using a small annotated dataset.

2.2 Artificially induced errors

The use of artificial data to train error correction systems has been explored by other researchers using a variety of techniques. Izumi, Uchimoto, Saiga, Supnithi, & Isahara (2003), for example, use artificial errors to target article mistakes made by Japanese learners of English. A corpus is created by replacing a, an, the or the zero article by a different article chosen at random in more than 7,500 correct sentences and used to train a maximum entropy model. Results show an improvement for omission errors but no change for replacement errors. Xie et al. (2016) add artificial errors for 2 of the most common error types in English article or determiner errors (They will generate and brainstorm ***the*** innovative ideas.) and noun number errors (Identification is becoming more important in our *society* → ***societies***.) which improves the BLEU recall performance significantly for the error types from 20.08 to 29.14 for article or determiner errors and from 31.50 to 51.00 for noun number errors. By adding these artificial errors, the overall $F_{0.5}$ performance of the model is improved 31.55 to 34.81.

Thus, artificial errors generally improve performance of corrections systems. However, there are 2 important factors when generating artificial errors. The first one is the resemblance of the error to human-made errors. If the error does not fit into human-made errors the artificial error is not useful, increasing the size of the training corpus, and thus making the training of the network more cumbersome. The second problem is the guarantee that the generated error is indeed an error. For instance, errors regarding verb tenses are impossible to generate because although you can change the tense of a specific verb it is difficult to guarantee the wrongness of the resulted sentence. Thus, only some specific error types can be produced. Moreover, this approach has the disadvantage that it has to be customized for each language.

2.3 Inducing noise with neural methods

Inspired by the backtranslation methods for machine translation (Xie, Genthial, Xie, Ng, & Jurafsky, 2018) generate grammatical erroneous sentences by encoding a clean source sentence, and decoding into a erroneous target. This approach needs a training corpus, the model being trained by inverting the source and the target of the corpora, thus clean→noisy pairs are used. In their work, they used a convolutional encoder-decoder architecture (Kalchbrenner et al., 2016) and a modified beam-search. However, they mentioned that any encoder-decoder, including those based on RNN networks should work in a similar manner.

The beam search is modified to generate more diverse sentences, and making sure that they are not grammatical correct sentences. The modifications imply adding different values to probabilities of the beam candidates, thus reordering them. The best method that worked for is called randomizing noising which implies adding $r * \beta_{random}$ to all the candidates in the beam search. β_{random} is a random variable sampled uniformly from [0, 1] interval, while *r* is a constant. For sufficiently large values of *r* this has the effect of randomly reordering the candidates.

3 METHOD

The current work has the final objective to combine, measure each individual impact of the 3 previously methods of augmenting an annotated dataset. As mentioned previously, a minimal annotated dataset (approx. 20k samples) is required to infer the distribution of the human made errors for Romanian. With no such corpus it is impossible to generate errors with the same distribution. Given the dataset it does not exists yet, the current experiments are limited, but their future use is based on the existence of such dataset.

3.1 Wikipedia edits

As presented below Wikipedia edits, filtered or unfiltered improve performance. However, most of the edits are content based, thus a further filtering will be applied based on a small dataset containing common mistakes. For now, all Romanian edits were extracted using the tool used by (Boyd, 2018), namely[1]. The Wikipedia extractor presented in (Grundkiewicz & Junczys-Dowmunt, 2014) which filters edits only minor modifications is also considered in the future work. Some example of Wikipedia edits are presented Table 1.

Table 1: Examples of Wikipedia edits.

Before editing	After editing
Este considerat a fi cel mai mare dramaturg român și unul dintre cei mai importanți scriitori români.	George Călinescu *îl considera* a fi cel mai mare dramaturg român și unul dintre cei mai importanți scriitori români.

[1] https://github.com/attardi/wikiextractor

Înainte de a-și publica pamfletul în broșură, Caragiale *a trimis* […]	Înainte de a-și publica pamfletul în broșură, Caragiale *trimisese* primul capitol […]
[…] iar în 1892 și-a exprimat intenția de a se *expatria la* […]	[…] și-a exprimat intenția de a se *exila la* […]
A publicat în revista literară *bimensuală* Convorbiri […]	A publicat în revista literară *bilunară* Convorbiri […]
A fost numit, *cu* decret regal […]	A fost numit, *prin* decret regal […]
[...] *a avut loc prima reprezentație* a piesei […]	[…] *premiera* piesei […]
În *aceste* împrejurări și-a manifestat calitățile *sale* de […]	În *acele* împrejurări și-a manifestat calitățile de […]
[…] *în presa vremii* […]	[…] in presa *contemporană* […]

As it can be noticed, some edits are not mistakes. For instance, verb tenses are changed because this the recommended style of writing in Wikipedia, but it may not be in another type of text. The edits that qualify as a mistake are a more subtle type of mistakes. For instance, the term "*bimensuală*" is replaced with the "*bilunară",* which its modern version and it is a change that you may want to suggest to a user, but is not necessarily a mistake. Some edits are just good paraphrasing. It is very difficult to appreciate how much rephrasing is required in a correction system, however one aspect is clear, Wikipedia edits are not very common, obvious mistakes. Thus, probably they should be used in a later phrase of the training process, when the learning rate is smaller.

3.2 Artificially induced errors

Artificial errors generation was most explored within this report. The generation of more complex artificial errors requires more advanced tools in the target language. Specifically, language-specific tags are required to generate errors such as: gender mismatch, noun number, case mismatch for nouns, etc.

A tool with such functionalities was developed because no such NLP library could be found. Based on the universal dependencies (UD) corpora developed (Barbu Mititelu, Ion, Simionescu, Irimia, & Perez, 2016) containing 218,511 tokens, 50 dependencies relations and

17 specific POS tags. The model used is the standard spacy neural model[2]. The accuracy of the model for tag prediction is *97.20%*. The model produces for each sentence a tokenization, and for each token a string including all POS tags and their values. The current tool constructs all possible enum values for each specific POS tag. For instance, the POS tag named *degree* there are three possible values: cmp (comparative, e.g. superior, superioară), pos (positive, e.g. mare, asemenea, mult) and sup (superlative, e.g. extrem, perfect). For POS tag named *gender* there are 2 possible values: fem (feminine) and masc (masculine). These enums are helpful in constructing artificial errors because given a word from a sentence you can request the modification of a specific POS tag value for a specific word, and the tool will provide the corresponding modified word. For instance, you can change the gender, number, definiteness article or case for a noun, the person, pronoun type or strength for pronouns or the tense, verb form or person for a verb. Moreover, we parsed all UD dataset and created a dictionary with all tokens and their associated POS tags features. Thus, when a request for a modified POS tag feature for a specific word is made, the tool searches for a matching word only in the previously created dictionary. Although the tool could have included predicted words in the dictionary, this solution was preferred because the predicted POS features for new words are not correct all the time, unlike the POS features for the words present in the UD corpus. Moreover, this tool can be later use to design features for the correction system.

The tool is integrated in the open source NLP toolkit Readerbench[3] and can be freely used by everyone. An example of all possible features encountered for the word lemma for the noun "prim" are shown below:

- primul:
    - RoCaseEnum: RoCaseEnum.ACC, RoCaseEnum.NOM (form of the noun)
    - RoDefiniteEnum: RoDefiniteEnum.DEF (definite form)
    - RoGenderEnum: RoGenderEnum.MASC (masculine form)
    - RoNumberEnum: RoNumberEnum.SING (number in singular form)
    - RoNumTypeEnum: RoNumTypeEnum.ORD (numeral type showing order, not comparison)

[2] https://spacy.io/api/cli#train
[3] https://git.readerbench.com/ReaderBench/readerbenchpy

    - RoNumFormEnum: RoNumFormEnum.WORD (numeral form described through a word, not an Arab or Roman digit)
- prima:
    - RoCaseEnum: RoCaseEnum.ACC, RoCaseEnum.NOM,
    - RoDefiniteEnum: RoDefiniteEnum.DEF
    - RoGenderEnum: RoGenderEnum.FEM
    - RoNumberEnum: RoNumberEnum.SING
    - RoNumTypeEnum: RoNumTypeEnum.ORD
    - RoNumFormEnum: RoNumFormEnum.WORD
- primii:
    - RoCaseEnum: RoCaseEnum.ACC, RoCaseEnum.NOM,
    - RoDefiniteEnum: RoDefiniteEnum.DEF
    - RoGenderEnum: RoGenderEnum.MASC
    - RoNumberEnum: RoNumberEnum.PLUR
    - RoNumTypeEnum: RoNumTypeEnum.ORD
    - RoNumFormEnum: RoNumFormEnum.WORD
- primei:
    - RoCaseEnum: RoCaseEnum.DAT, RoCaseEnum.GEN,
    - RoDefiniteEnum: RoDefiniteEnum.DEF
    - RoGenderEnum: RoGenderEnum.FEM
    - RoNumberEnum: RoNumberEnum.SING
    - RoNumTypeEnum: RoNumTypeEnum.ORD
    - RoNumFormEnum: RoNumFormEnum.WORD

The list is not exhaustive, continuing with word forms such as: primilor, prim-, primelor, prim, primă, prime, etc.

Some examples for the the verb to „see” (ro. „a vedea”) are listed below:

- văzuse:
    - RoMoodEnum: RoMoodEnum.IND
    - RoNumberEnum: RoNumberEnum.SING (singular)
    - RoPersonEnum: RoPersonEnum.THIRD (third person)

    - RoTenseEnum: RoTenseEnum.PQP (past perfect tense equivalent in Romanian)
    - RoVerbFormEnum: RoVerbFormEnum.FIN (general verb form)
- văzând:
    - RoVerbFormEnum: RoVerbFormEnum.GER
- văzut:
    - RoGenderEnum: RoGenderEnum.MASC
    - RoNumberEnum: RoNumberEnum.SING (singular)
    - RoVerbFormEnum: RoVerbFormEnum.PART (general verb form)

The left side of each white bullet represents a shorthand for the specific POS tag, while on the left side are the values that match with the given word form. Not all POS feature exists for each word. For instance, there is no purpose of a verb tense POS tag for a noun, thus, we can consider that nouns have no verb tense POS feature. As can be noticed from above, there can be multiple values for a specific POS tag (RoCaseEnum for word "primii" can have 2 values: ACC ("acuzativ") and NOM ("nominativ"), meaning that the word form matches both cases. However, in most cases, each POS tag has either one or zero values.

For the current work we defined some common mistakes that occur specifically for Romanian language. These errors are not limited to grammatical errors, but include also spelling and orthographic errors. The errors types are defined in Table 2.

Table 2: Mistakes types for Romanian language.

Mistake type	Description	Example
Tp	Typos qwerty	Am mers la [**mahazin** -> magazine].
Diac	Missing/extra diacritics	[**Scoala** -> Școala] din zare.
Vt	Verb tense	[Merg -> Am mers] la magazin ieri].
Vf	Verb form	Masinile sunt **[prevazut** -> prevazute] cu suspensii.
SVA	Subject-verb agreement	El [**mergem** -> merge] la magazin.

Mistake type	Description	Example
Pre	Prepositions	Acest țel oarecum a început să piară [**ca și** -> ca] importanță. Spre unul [**din** -> dintre] directorii firmei.
Vir	Missing/extra commas	[**Astfel** -> Astfel,] s-a demonstrat teorema. Soră-mea cea mare [**, ->**] m-a ținut odată în casă
FC	Spelling	[**Copii** -> Copiii] merg la scoala prea mult. De mult [**vroiam ->** voiam]. 1000 de persoane care ar putea [**știi ->** sti**]** ceva.
Cr	Extra/missing dash	[**Dintro** -> Dintr-o] greseala am plecat.
Sp	Spacing	**deasemenea** vs de asemenea, numai vs numai
Gen	Gender	Fata [**frumos** -> frumoasă] a plecat.
Case	Noun case	Am iesit la [**pensii** → pensie]
PI	Strongness form of pronoun	Ea [**însumi** -> însăși] a câștigat.
AcP	Owner agreement	Băiatul [**al** → a] cărui carte
Cap	Capitalization errors	[**ioana →** Ioana**]** merge la magazin.

For each sentence only one mistake type is present only once. The mistakes are generated as follows:

- Typos qwerty: typos generated by using Levenstein distance to find permutations with respect to qwerty keyboard distribution implemented in[4]. Characters which are present in Romanian and not in English are added in the keyboard layout.
- Missing/extra diacritics: one random word is selected, and one diacritic is added or deleted.
- Verb tense: this error type was not generated because of changing the verb tense does not guarantee the wrongness of the sentence

[4] https://github.com/wsong/Typo-Distance/blob/master/typodistance.py

- Verb form: there are 4 possible verb forms: (“Gerunziu”, “Infinitiv”, “Participiu”, “Supin”). The verb form is changed. These 4 verb forms are used in very different context, thus changing it consistently produces a grammatical mistake.
- Subject-verb agreement: the conjugation of the verb is changed.
- Prepositions: a pre-defined dictionary containing regular expressions with the most common preposition mistakes is used to replace/add/remove prepositions with the wrong ones.
- Missing/extra commas: commas are simply removed or added.
- Spelling: this category contains spelling mistakes, but unlike *Tp* category (typos qwerty) contains common spelling mistakes that people make. The mistakes are generated using a predefined dictionary with regular expressions, similarly with prepositions category.
- Extra/missing dash: Dash (“cratimă”) is a very common mistake in Romanian. Changing a word by adding or removing its dash always produces a mistake, because for a given context only one type of word (with/without dash) fits. The mistakes are generated also using a predefined dictionary with regular expressions for elimination/addition of dashes.
- Spacing: some common mistakes imply missing/extra space between words. There are some typical mistakes in Romanian regarding spacing which are predefined in a dictionary with regular expressions.
- Gender: The gender of adjective or nouns is changed. For adjectives this always produces a mistake, for nouns the mistake is not always guaranteed.
- Noun case: a noun is changed to be in a different case, making sure that the word form is different in that case from the initial one. This generally produces a mistake because the cases are used in very different contexts. However, the mistake is not always guaranteed.
- Strongness form of pronoun: a specific form of pronoun (“de întărire”) is commonly mistaken in Romanian. The pronoun form is changed with a random different one from a dictionary.

- Owner agreement: another common mistake in Romanian, the agreement is changed with a another random one.
- Capitalization errors: uppercase letters are changed to lowercase.

The corpus used to generate mistakes contains copyright free books such as The Count of Monte Cristo by Alexandre Dumas. A corpus with more modern terms, such texts from news outlets is considered for future work.

3.3 Inducing noise with neural methods

The idea of noising is mainly inspired from (Xie et al., 2018) which modifies an encoder-decoder architecture and train on a reversed corpus. In their experiments they used a convolutional encoder-decoder architecture, but they mentioned that similar results should be expected using a normal recurrent based encoder-decoder.

Thus, for this report a vanilla encoder-decoder at character level based on LSTM (Hochreiter & Urgen Schmidhuber, 1997) was implemented and trained it be reversing the source and the target. At each step in the decoder a dense layer using softmax is applied on the output of the LSTM decoder. The loss used is categorical cross entropy. The optimizer used is *ADAM* (Kingma & Ba, 2014). The dimension for the LSTM cell was 128, the embedding for each character was trained, having 28 dimensions. The batch for a training step was consists of 32 examples. After each epoch all entries from the training set shuffled. The method used for training was teacher forcing which consists of predicting the next word given all correct words that should have been predicted (and not that were predicted during training) in addition to the input words. A beam search algorithm was also implemented using a beam of 8 with and without random noising. The total number of the parameters is thus approx. 184k parameters. The model can be simplified with Figure 2:

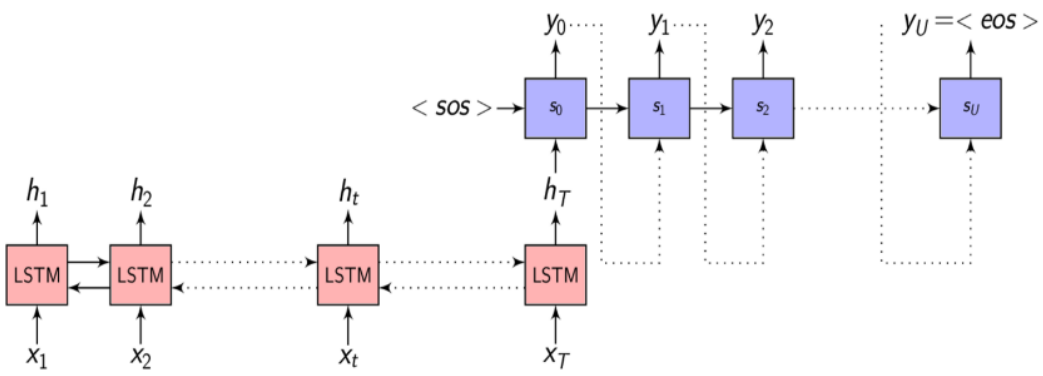

Figure 2. Encoder model.

The decoder hidden and cell state are initialized with the last hidden and cell state of the encoder. The character level was chosen to avoid out-of-vocabulary words, but also to avoid the big number of parameters. Moreover, Romanian being a rich morphological language, character level embeddings may work even better than in English where they already been applied to this problem.

As an experiment, the model was trained on Lang-8 corpora[5] which contains approx. 100k entries in English. Because this is a neural method, it requires lots of examples to be trained, thus it will be trained with Romanian examples after the previous 2 methods of generating mistakes are applied (artificial mistakes and filtered Wikipedia edits). Moreover, because learning to produce the exact mistake given a clean sentence is not really the purpose of this model, the only evaluation that can be done of the model is to use the generated corpus for another established for this task, and compare the results.

Unfortunately, the model could not be trained enough to produce results. After 5 epochs of training and achieving a loss of approx. 0.3, the model failed to just copy characters, producing mostly noisy output.

The advantage of this architecture is that it can be used to also correct sentences, by only modifying the training process and eliminating the beam search noising mechanism. In fact in (Xie et al., 2018) they used the exact same architecture to train a model for corrections.

4 RESULTS FOR DETECTING AND CORRECTING WORD LEVEL ERRORS

Several experiments focusing on the correction of mistakes at word level were conducted. For these experiments, 2 new corpora were also generated in the manner presented below.

The corpus that was generated contains sentences extracted from some Romanian books, with one word altered. We constructed 2 corpora, each with its own alteration type of word. The first one contains word altered in a typo manner (typos), specifically qwerty common errors, while the second one (inflected) contains grammatical errors, by changing the form of the word. The form of the word was changed by finding the most similar word in fastText

[5] https://sites.google.com/site/naistlang8corpora/

(Joulin et al., 2016) which has the same lemma. Other approaches for generating an inflected corpus are to be tried. Samples which contain at least one content word which is not in the fastText (Joulin et al., 2016) dictionary were eliminated. Namely, we took a dictionary with all inflected words. The statistics for the 2 corpora are presented Table 3:

Table 3: Generated corpora statistics.

corpus	sentences/samples	Dictionary (# words)	size (MB)
inflected	650k	150k	150MB
Typos	550k	150k	152MB

4.1.1 Models for correcting word level mistakes

The model proposed for now are straight-forward. For each dataset there are 2 BiGRU passed to each window of words/characters. The first one passes through the fastText word embeddings, while the second one passes through the window of 29 characters embeddings centered at the word for which the prediction is made. The embeddings for characters are learnt, while for word are taken pretrained. The last hidden state of the BiRNNs are concatenated and then passed through a dense layer. The a softmax layer over a dictionary is used, the output being a word. Experiments were conducting using the concatenation of the word embedding of the altered word to the 2 hidden states output. The resulted are presented in the Table 4.

Table 4: Results for correction at the word level.

Task	Model	Epochs	Val Accuracy
inflected	word	33	0.519
inflected	char-GRU	46	0.518
inflected	word + sent-GRU + char-GRU	15	**0.610**
typos	word	17	0.622
typos	char-GRU	35	0.733
typos	word + sent-GRU + char-GRU	18	**0.738**

All hyperparameters of the models are similar, presented in the table below:

- batch size: 64
- window characters (for BiGRU): 29
- window sentence (for BiGRU): 30 (contains the entire sentence all the time)
- BiGRU cell size: 64
- dense size: 128
- embedding characters (trained): 28
- embeddings words size (pre-trained): 300

The relatively small difference in validation accuracy between the simple word model and the more complex ones (which takes into account the context) show that future work which takes better advantage of the context will increase the accuracy.

4.1.2 Models for detecting word level mistakes

For detecting misspelled words, we used a similar model, with 3 branches concatenated and then passed through 2 dense layers. The hyperparameters which were changed are presented below:

- dense sizes: 128, 64
- dropouts: 0.2, 0.1

The results for these models are as presented in Table 5.

Table 5: Detection statistics.

Task	Model	Epochs	Val Accuracy	Precision	Recall	$F_{0.5}$
inflected	word	25	0.820	0.782	0.843	0.794
inflected	word + sent-GRU + char-GRU	15	**0.927**	**0.909**	0.943	0.916

The spelling correction systems are mainly focused on having a higher precision accuracy, not necessarily recall, this is why we reported $F_{0.5}$ score and not F_1 or higher F_s. I have also tried to improve precision by taking a prediction meaning (misspelled word) to be positive if the probability of the corresponding class is much higher than 0.5 (namely 0.6, 0.8, 0.9). Thus, the

results for these values are as presented in Table 6 (for the word + sent-GRU + char-GRU) model.

Table 6: Classifying by different threshold.

Threshold	Precision	Recall	$F_{0.5}$
0.50	0.909	**0.943**	0.916
0.40	0.926	0.933	0.928
0.25	0.947	0.915	0.940
0.10	**0.971**	0.879	**0.951**

Table 7: Examples from the detection system (for threshold 0.5). Text in bold marks incorrect initial text

Sentence	Type
— vrea/**vrem** oare old shatterhand să mă abată de la datorie ?	tp
vreau și eu pedepsirea/**pedepsire** ucigașilor .	tp
datinile îmi poruncesc să rămîn lîngă morții mei/**meu** pînă vor fi înmormîntați .	tp
— și cînd va/**vruseră** fi înmormîntarea ?	tp
pentru că nu pot/**putea** să - l urmăresc personal , sarcina va reveni altcuiva .	tp
Acum/**acu'**, cînd era vorba de chestiuni concrete , winnetou deveni calm ca de obicei.	tp
— acesta e numele lui/**el** adevărat ? el lui	tp
— aveti/**avuseși** cunoscuți prin apropiere , poate în vreun fort ?	tp
— voiam/**voi** să ... să ne luăm după ...	tp
- am/**avea** ghicit însă gîndul și l - am întrebat :	tp
sau aveati/**avea** de gînd să - i atacați la întoarcere și ...	tp
trecuse mult de amiază și sam mă întîmpină intrigat/**intrigată**	tp
se lăsă/**lăsa** o tăcere adîncă .	tp
apoi v - ați ascuns în **pădurice** și ne - ați observat de susul , din copaci . pădurice	fp
sper că voi răspunde cu cinsteo **încrederii** pe care fratele meu roșu mi - o acordă **.**	fp
gîndul **acesta** îmi grăbi pasul .	fp
n - avea izbutit din primul moment , pămîntul **fiind** aici încă prea dur .	fp
cei trei **cai** se mai afla în pădurice .	fp
trecuse mult de amiază și sam mă întîmpină intrigată :	fp
avea , nu **zic** ba .	fp
zarvă nu duce la nimic	fp
gîndul acesta îmi grăbi pasul**/pasii** .	fn
în loc de răspuns , i - am chemat pe apasi/**apaș** la mine .	fn
așadar , ce aduce tovărășia mea : viață sau moarte/**morții**	fn
Draga/**dragă** de ea , frumoasa și buna fată indiană	fn
— mai tacă/**tăcu** - ți gura cu greenhorn - ul dumitale și lasă glumele !	fn
santer poate să treacă fie peste munți/**munte** , fie printre munți , cum îi convine .	fn

5 FUTURE WORK

For future improvements on the neural generation architecture, an attention mechanism should be implemented in order to improve the training process. In the current experiments, the model had problem even learning to copy characters from the source to the target sentence, needing more than 5 epochs with the entire corpora (100k examples) to learn this. As mentioned in (Xie et al., 2016) an attention mechanism was paramount importance for the performance of the model.

Although the research for word level corrections seems encouraging, the situation is not the same for general corrections implying more than one word. The main focus in the next report will be the finalization of corpus generation system, including a base corpus of at least 20k entries human-made. Part of the human-made corpus will represent the evaluation corpus for the entire work.

BIBLIOGRAPHY

Barbu Mititelu, V., Ion, R., Simionescu, R., Irimia, E., & Perez, C.-A. (2016). The romanian treebank annotated according to universal dependencies. *Proceedings of The Tenth International Conference on Natural Language Processing (HrTAL2016)*.

Boyd, A. (2018). Using Wikipedia Edits in Low Resource Grammatical Error Correction. *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-Generated Text*, 79–84.

Dahlmeier, D., & Ng, H. T. (2012). Better evaluation for grammatical error correction. *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 568–572. Association for Computational Linguistics.

Grundkiewicz, R., & Junczys-Dowmunt, M. (2014). The WikEd error corpus: A corpus of corrective Wikipedia edits and its application to grammatical error correction. *International Conference on Natural Language Processing*, 478–490. Springer.

Grundkiewicz, R., & Junczys-Dowmunt, M. (2018). Near human-level performance in grammatical error correction with hybrid machine translation. *ArXiv Preprint ArXiv:1804.05945*.

Hochreiter, S., & Urgen Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Izumi, E., Uchimoto, K., Saiga, T., Supnithi, T., & Isahara, H. (2003). Automatic error detection in the Japanese learners' English spoken data. *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, 145–148.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016). Fasttext. zip: Compressing text classification models. *ArXiv Preprint ArXiv:1612.03651*.

Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A. van den, Graves, A., & Kavukcuoglu, K. (2016). Neural machine translation in linear time. *ArXiv Preprint ArXiv:1610.10099*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*.

Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., & Ng, A. Y. (2016). Neural language correction with character-based attention. *ArXiv Preprint ArXiv:1603.09727*.

Xie, Z., Genthial, G., Xie, S., Ng, A., & Jurafsky, D. (2018). Noising and denoising natural language: Diverse backtranslation for grammar correction. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 619–628.