Review Article

Current Trends in the Use of Machine Learning for Error Correction in Ukrainian Texts

Rostyslav Fedchuk[1], Victoria Vysotska[1]

1. Lviv Polytechnic National University, Ukraine

The article's authors have provided a detailed problem description of identifying and correcting errors in Ukrainian-language texts. This paper provides a detailed analysis of the latest research and publications aimed at solving the problems of identifying and correcting errors in Ukrainian-language texts. The analysis of modern tools related to error correction in texts is presented along with a comparative description. The existing data corpora for the Ukrainian language were investigated to ensure they are relevant to solving GEC tasks. The need to create a large annotated data corpus, which will be prepared by a special team with linguistic expertise, was discovered. The opportunities, advantages, and disadvantages of modern machine learning models that interpret the task of detecting and correcting errors in texts as classification or machine translation were analysed. The need to develop a machine-learning algorithm that will take into account the specifics of morphologically complex languages, such as Ukrainian, was introduced. The work of the modern models was demonstrated, and screenshots were provided. The need for further research in the Ukrainian segment of machine learning to solve the problems of correcting errors in texts using various methods and approaches was revealed.

Corresponding authors: Rostyslav Fedchuk, rostyslav.b.fedchuk@lpnu.ua; Victoria Vysotska, victoria.a.vysotska@lpnu.ua

Introduction

Knowledge of the Ukrainian language is an indicator of education and patriotism, but it is quite important not only to know it but also to use it correctly. This became especially important and key

precisely with the beginning of a full-scale invasion, since through typical errors in texts it is possible to detect fakes or propaganda from the enemy. The problem of using the Ukrainian language to create textual content is the presence of errors that may occur during the writing, editing, and publishing of texts. These errors can be different: from spelling and grammar to stylistic. Even the most experienced authors sometimes make inaccuracies, which can affect the understanding of the text and cause misunderstandings among readers.

The relevance of this problem becomes especially important in the context of the large amount of information published online and offline. The growing number of websites, blogs, social media, and other sources of information leads to the need for quality and accuracy in published texts. In addition, the Ukrainian language as a national resource is of great importance for the preservation of cultural heritage and the development of Ukrainian society.

Errors in texts can become an obstacle to the correct perception of information, which negatively affects communication and understanding of important issues. This can affect the level of trust in the source of information, reducing its authority and influence.

Formulation of the problem

The beginning of the era of artificial intelligence opened new horizons for automatic natural language analysis and processing. Natural Language Processing (NLP) is a branch of artificial intelligence that deals with the understanding, interpretation, and generation of human language by computer systems. The main goal of NLP is to make machines able to analyse, understand, and reproduce human language in the same way that humans do. To achieve this goal, NLP includes a wide range of methods and techniques for text pre-processing, including segmentation, tokenization, lemmatization, and syntactic, discourse, or semantic analysis.

NLP is used in various fields such as search engines, chatbots, machine translation, social media sentiment analysis, fraud detection, automatic text processing, etc. Using NLP to identify and correct errors in texts is one example of its application.

Grammar Error Correction (GEC) is the process of detecting and correcting grammatical errors in text using software or machine learning algorithms. The main purpose of GEC is to detect and correct spelling, syntactic, and stylistic errors that can affect the understanding of the text and its quality. As of today, most research on solving GEC problems is focused on error correction in English-language texts [1].

The GEC task for the Ukrainian language is an open challenge for many reasons. Firstly, due to the morphological complexity of the language, existing algorithms and machine learning models for the English language are not ideal candidates for use in error correction in Ukrainian-language texts. Secondly, there is still a small amount of research, tools, or NLP corpora for solving the GEC problem for the Ukrainian language. Third, manual data annotation requires a lot of effort by professional linguists, which makes the creation of text corpora a time- and resource-consuming process. The biggest difficulty is that the Ukrainian language is resource-poor, and to date, there is only one annotated GEC dataset [2] and few high-quality pre-trained transformer models compared to the English language. In addition, the Ukrainian language contains many exceptions and does not have a clear word order. All this greatly complicates language analysis.

Analysis of recent research and publications

Starting from the 80s of the 20th century, society began to develop the first systems for identifying and correcting errors in texts. Such systems were based on rules [3].

Rule-based methods for correcting errors in text are based on defined rules that account for common grammatical and spelling errors. These may include spelling rules, grammar rules, stylistic rules, etc. After creating a set of rules, the text is analysed according to these rules to detect errors. In some cases, there may be a need for recursive text analysis when correcting one error may cause others to appear.

The advantage of rule-based methods is their transparency and the ability to precisely control the error correction process. However, these methods may be limited in that they may miss complex or contextual errors that do not follow established rules. This method requires a very large number of rules to work effectively with various types of text and therefore requires knowledge of linguistics. Therefore, the main drawback of this approach is the impossibility of covering all possible errors with rules.

Syntax-based methods for text error correction use structural properties of language, such as grammar and syntactic relationships between words, to detect and correct errors. This method uses parsers that assign a tree-like structure to each sentence. It is important to identify syntactic structures in the text, which may include identifying the parts of speech of each word, analysing the structure of the sentence, determining the relationships between words and phrases, etc.

Advantages of syntax-based methods include their ability to detect grammatical and syntactic errors that are not always detected by other methods. However, the syntactic approach has a major problem: it

requires a complete grammar that covers all types of texts. To date, there is still no reliable parser with wide coverage available in the public domain. In addition, parsers suffer from the inherent ambiguity of language, so it's common to return more than one result even for correct sentences.

Statistical methods for correcting errors in the text are based on the analysis of the statistical properties of the text to determine the probability of matching a specific word or phrase with the standard correct version. Using statistical methods requires a large corpus of valid texts on which the model will be trained. For each word in the text, the probabilities of its correction, replacement, deletion, or insertion are calculated based on the probabilities defined in the model. For example, a word with a low probability can be removed or replaced with a more likely word. After calculating the probabilities for different error correction options, the one with the highest probability is selected.

Statistical methods allow efficient correction of errors in text, especially if a large variety of correct texts is available for training the model. However, these methods may also have limitations in correcting complex or contextual errors that are not reflected in the statistical properties of the text.

Machine learning methods for correcting errors in the text are based on various mathematical and statistical concepts and algorithms that allow computers to predict the likely correction of the text [4]. While predictive algorithms are very important, collecting a large amount of data that will be used to train and test the model is equally important. Data can be collected from various sources such as the internet, books, news, etc. The data is then subjected to pre-processing, including segmentation, tokenization, lemmatization, de-noising, removal of unnecessary characters, etc.

Advantages of using machine learning methods include their ability to adapt to different types of errors and text contexts, and their ability to automatically learn from new data to continuously improve performance. However, these methods require large amounts of training data and can be costly in terms of computing resources.

One solution is to treat GEC as a classification task. This means that a system that works with GEC can treat each piece of text as an individual example and classify it based on whether it contains grammatical errors or not. The basic idea is that the system receives text as input and returns a classification of each fragment of this text with further instructions.

On the other hand, **the GEC task can be considered as a machine translation (MT) task**. To solve the GEC problem using Machine Translation, we can use similar approaches and models that are used to translate text from one language to another. The translation model is trained on pairs of sentences where one

sentence is "wrong" and the other is "correct" and learns the correspondences between them. Interpreting GEC as Machine Translation opens up the possibility of using a wide range of methods and techniques developed for machine translation.

Statistical Machine Translation (SMT) is an approach to machine translation that uses statistical models to translate text from one language to another [5]. The basic idea is to build a model that can produce the most likely translation based on the statistical analysis of a large number of parallel texts (texts containing the same content but written in different languages).

The conducted studies CoNLL-2013 [6] and CoNLL-2014 [7][8] showed the effectiveness of using the trained SMT model to correct various types of errors. However, such models become less effective when considering contexts for a long time.

Statistical machine translation was one of the popular approaches to machine translation before the advent of neural networks. Although it is still used in some situations, many newer models of machine translation have proven to be more effective.

In studies [9][10], **NMT (Neural Machine Translation)** was presented for the GEC task, which uses deep neural networks for automatic text translation, that is, for modelling the mapping between input and output texts. The main components of NMT are the encoder, decoder, attention mechanism, and loss function.

The main advantage of NMT is that it can perform error correction in context, that is, it takes into account the semantics and grammar of the text when generating the correction. This helps ensure better quality and more natural translations, especially in complex or low-resource language pairs, such as Ukrainian. NMT has proven to be very effective for many machine translation tasks and is now widely used in many translation systems and other areas of natural language processing.

Recurrent neural networks (RNNs) are a class of neural networks that are designed to work with sequential data such as text or time series. One of the key features of RNNs is their ability to remember and use information from previous steps to process the current input value. One of the main problems of RNNs is the occurrence of the problem of vanishing and fading gradients when training on long sequences. This can result in the model not being able to effectively account for remote dependencies in the data.

To solve the problem of falling RNN gradients, a new class of models, transformers, was proposed in [11].

Transformers are a deep learning architecture that has gained wide popularity in recent years, especially in natural language processing. Transformers are based on the attention mechanism and are an effective tool for processing parallel data. This enables transformers to better model long-term dependencies in text and to achieve better results on many natural language processing tasks.

The main advantages of transformers are as follows:

- Versatility: does not require significant changes in the architecture.
- Context-aware: considers the context of each word in a sentence during processing.
- Computational parallelism: allows processing of large amounts of data.
- Ability to learn long dependencies: able to learn long dependencies in the text thanks to the attention mechanism [12], which allows them to take into account the entire decision-making context.
- The possibility of pre-training on large data sets: increases the efficiency of the model.

The transformer consists of two parts: an encoder and a decoder (Fig. 1).

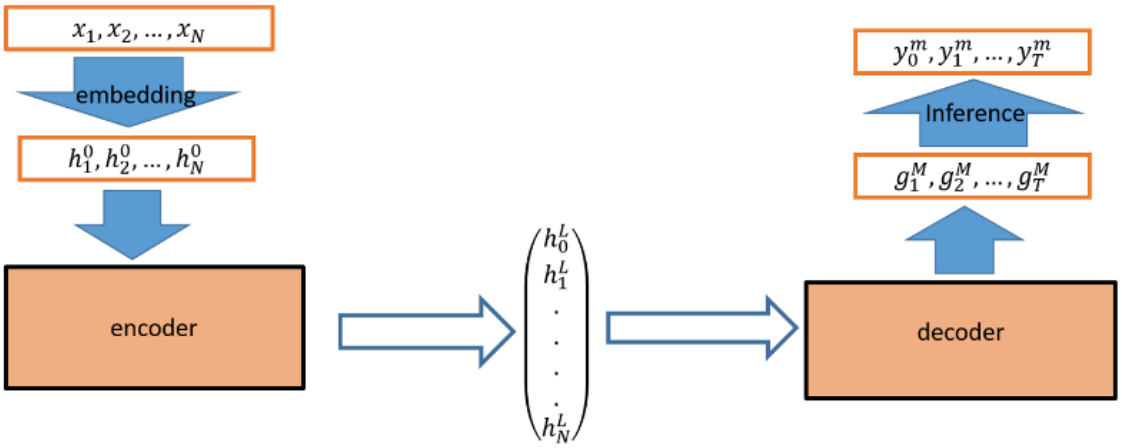

Fig. 1. *Transformer architecture* [4]

Let's look at an example of how the **encoder** works for the text "This sentence is misspelt". For simplicity, let's assume that we use a simple transformer with one layer of the encoding stack.

1. Tokenization: First, the text will be divided into separate tokens. In this case, we can split it into the following tokens: ["This", "sentence", "is", "misspelt"]. Also, the input sequences are preprocessed, and the prepared input data is used to train the sequential model. The trained streaming model is then used to predict the output for new input sequences. A very common algorithm is to first pre-

train the model weights on a larger synthetic dataset and then train them on a smaller set of GEC-specific texts in parallel. The derivation stage produces the correct final output sequence.

2. Embeddings: Each token will be converted into a vector representation (embedding). This can be, for example, one-hot encoding or a vector representation based on a previously trained word embedding. For example, "This" can be a vector [0.2,-0.3,0.7,...], "sentence" - [0.1,0.5,-0.2,...], and so on.
3. Submission to the encoder: Each token with its embedding is submitted to the input of the encoder. An encoder performs a sequence of operations, including attention mechanisms and neural layers, to process each token and its context in the input sequence.
4. Encoder output: At the encoder output, we get a sequence of internal representations of each token after processing by the encoder. These representations contain contextual information about each token in the input text. For example, the vector for "This" can look like [0.1,0.3,-0.2,...], for "sentence" - [0.2,-0.1,0.5,...], and so on.

The Attention Mechanism in transformers allows the model to focus attention on different parts of the input data (text) during processing. This mechanism consists of three main components: query (query), key (key), and value (value). Here's how it works:

1. Query is a vector that represents the current token (word or symbol) in the output sequence generated by the decoder (or the whole representation of the input token in the encoder). The query is used to calculate the similarity between this token and all other tokens in the sequence.
2. Each token in the input sequence (or encoder output) is also assigned a key and a value. The key is used to calculate the similarity between the current query and other tokens, and the value is the information associated with that token.
3. For each token in the input sequence (or encoder output), a "complex" attention is calculated, which takes into account the similarity between the current query and the key. This calculation is usually performed using a similarity function such as the dot product or cosine similarity. This attention is then normalized to obtain an attention distribution that sums to unity.
4. Finally, the model response is formed as a weighted sum of values, where the weights are determined by the calculated attention. This allows the model to pay more attention to important parts of the sequence and less attention to less important parts.

The decoder in the transformers, in turn, is used to generate sequences based on the internal representation of the input data that was created by the encoder.

Initially, the decoder receives an input internal representation that was created by the encoder. This internal representation contains information about the context of the source sequence that was used when processing the input data. Like the encoder, the decoder also uses positional embeddings to convey information about the position of each token in the sequence. This allows the decoder to take word order into account when generating the output sequence.

During the generation of the source sequence, the decoder uses self-similarity masks so that each token in the source sequence does not have the opportunity to "look" at future tokens. This helps ensure sequence generation is correct.

The decoder also uses an attention mechanism to focus on important parts of the input representation when generating the output sequence. This helps the decoder to consider the context of the input when generating each token of the output sequence.

After the decoder has generated the output sequence, the loss function compares it with the actual output sequence. Based on this comparison, the parameters of the model are updated to improve its quality.

In the end, the decoder generates the output sequence of tokens, one by one, using the attention mechanism and other mechanisms to take into account the context of the input data. Each subsequent token is generated based on the previous one and the context of the input. This process continues until the final length of the output sequence is reached or until a special termination character appears.

Formulation of the purpose of the article

The purpose of this work is the analysis and detailed examination of modern trends in the use of machine learning for the identification and correction of errors in texts in the Ukrainian language. To achieve this goal, the following research tasks were defined:

- Analysis of the problems of identification and correction of errors in Ukrainian-language texts.
- Analysis of the latest research and publications aimed at solving the problem of identifying and correcting errors in Ukrainian-language texts.
- Analysis of modern tools related to the correction of errors in texts and their comparative characteristics.

- Research of existing data corpora for the Ukrainian language that are relevant to solving GEC problems.
- Research of the possibilities of applying artificial intelligence methods, neural networks, and machine learning algorithms to improve the quality of automatic text correction and adapt the system to new templates and text structures.

The object of the study is the process of identifying and correcting errors in Ukrainian-language texts based on Ukrainian corpora using various machine-learning models.

The subject of the study is machine learning models created to solve the problems of identification and correction of errors in Ukrainian-language texts learned from Ukrainian-language data corpora.

Available corpora of the Ukrainian language

Machine learning algorithms have become an important component for solving various tasks; their capabilities in forecasting, classification, and data processing make them a powerful tool for solving complex problems. However, it is important to remember that the quality and effectiveness of machine learning algorithms largely depend on the quality and representativeness of the data they were trained on. This is where corpora and data sets play a role.

Corpora are collections of text or other data that are used to train, test, and validate machine learning models. They can be collected from a variety of sources, including the Internet, books, scholarly articles, social media, and more. Corpora enable machine learning algorithms to learn patterns in data and improve their predictive and analytical abilities. They can also be used to generate new data, for example, to develop and improve synthetic language models.

However, corpora are only part of the data ecosystem. Before processing, corpora often require cleaning, annotation, and other preparation methods to ensure high quality and representativeness.

As of now, there is already a considerable number of corpora for machine learning, designed specifically for the Ukrainian language. These corpora have become a key tool for the development and improvement of machine learning models in the context of the Ukrainian language. However, compared to corpora for the English language, there are still not many resources for the Ukrainian language, and they do not cover as many areas of NLP as we would like.

Due to the lack of a large free corpus, in [13] the authors created UberText 2.0. At the moment, UberText 2.0 is one of the largest corpora for the Ukrainian language, having 3.274 billion tokens, consisting of 8.59

million texts, and occupying 32 gigabytes. In addition to text, UberText 2.0 includes text normalization, speech detection, sentence segmentation, tokenization (with punctuation preservation), lemmatization, and POS tagging.

Below are some of the most famous machine-learning corpora for the Ukrainian language:

- BRUK – a corpus of the modern Ukrainian language [14], 1 million words.
- GRAC – a reference corpus of the Ukrainian language compiled by hand [15].
- OSCAR – data extracted from Common Crawl, 28 GB [16].
- Zvidusil – a web corpus with syntactic annotation [17].
- KUM – text corpus of the Ukrainian language [18].
- ukTenTen – Ukrainian corpus from the Web, 7.5 billion tokens.
- UberText 2.0 – news, Wikipedia, social, fiction, and legal literature.
- Malyuk – a collection of OSCAR, UberText 2.0, and Ukrainian News corpora, 113 GB of text.
- CC-100 – documents pulled from Common Crawl.
- mC4 – data from Common Crawl, 196 GB.
- Ukrainian Twitter corpus – Ukrainian Twitter corpus for detecting toxic text.
- Ukrainian forums – 250,000 sentences collected from forums.
- Ukrainian news headlines – news headlines, 5.2 million.
- OPUS
- Polish-Ukrainian Parallel Corpus
- Back-translated monolingual Wiki data
- Wiki Edits – 5 million sentences from the history of Ukrainian Wikipedia edits.
- ZNO – 4000 questions and answers.
- UA-GEC – an annotated GEC corpus.
- NER-uk – an annotated BRUK for entity discovery (NER) problems.
- Yakaboo Book Reviews – an annotated corpus of book reviews, ratings, and descriptions.
- ua-news – annotated corpus of news.
- UA-SQuAD – an annotated corpus of data on answers to questions.
- WSC Dataset – an annotated corpus of manual translation.
- VESUM – a dictionary of POS tags.
- obscene-ukr – dictionary of profanity.

- Word stress dictionary – stress for 2.7M word forms.
- Heteronyms – a set of homonyms.

However, as of today, there is only one corpus that is designed specifically for error correction for the Ukrainian language - UA-GEC (Fig. 2).

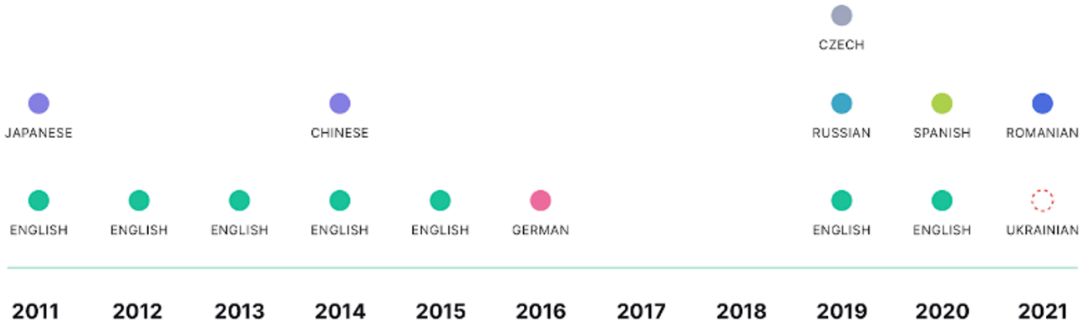

Fig. 2. *GEC datasets for different languages.*

UA-GEC was created in 2021 and contained 1011 texts with errors totaling 20,715 sentences [9]. These texts have been collected from various sources, such as chats, articles, posts from social networks, essays, or even letters in a formal style. Professional editors analyze texts for errors related to fluency, grammar, punctuation, and spelling.

Already in 2022, UA-GEC was updated to the second version, which already includes twice as many corpora, namely 34,000 sentences [19]. The categories of grammar and stylistics have been separated, which makes it possible to separately correct both grammatical errors and the use of incorrect style, which is a rather difficult task. Error categories include punctuation, spelling, grammar, and style (Fluency). In total, the Grammar category was increased by 13 subcategories, and the Style category by 5.

Spelling was recorded to account for 19% of corrections, while punctuation corrections occur more frequently due to strict punctuation rules in the Ukrainian language.

The GEC+Fluency corpus includes data from 828 authors, 1,872 texts, 33,735 sentences, and 500,618 tokens with an error rate of 8.2% for the entire corpus (Figure 3).

Split	Writers	Texts	Sentences	Tokens	Annotations	Error rate
Train	752	1,706	31,038	457,017	38,383	8.1%
Test	76	166	2,697	43,601	7,865	9.0%
TOTAL	828	1,872	33,735	500,618	46,248	8.2%

Fig. 3. *Statistics of the GEC+Fluency corpus* [19].

The corpus is suitable for the development and evaluation of GEC systems in the Ukrainian language, as well as for the study of multilingual and low-resource NLP, morphologically rich languages, and document-level GEC, including language fluency correction.

Below you can see the statistical data of the Ukrainian GEC corpus in comparison with the corpora of other languages (Fig. 4).

Language	Corpus	Sent.	Er.
English	Lang-8	1,147,451	14.1
	NUCLE	57,151	6.6
	FCE	33,236	11.5
	W&I+L	43,169	11.8
	JFLEG	1,511	
	CWEB	13,574	1.74
Czech	AKCES-GEC	47,371	21.4
German	Falko-MERLIN	24,077	16.8
Romanian	RONACC	10,119	
Russian	RULEC-GEC	12,480	6.4
Spanish	COWS-L2H [3]	12,336	
Ukrainian	**UA-GEC**	**33,735**	**8.2**

Fig. 4. *Comparative statistics of the UA GEC corpus with other languages* [19].

doi.org/10.32388/N4VGBJ.2

Natural language processing tools for Ukrainian

As of now, all existing software tools for solving GEC problems in the Ukrainian language use the Rule-based approach. One such program is LanguageTool [20], which can check grammatical, stylistic, and spelling errors in texts written in many languages, including Ukrainian. However, there is a specific project to demonstrate the LanguageTool API for the Ukrainian language - NLP-UK [21]. This project contains tools for normalization, cleaning, tokenization, lemmatization, and POS tagging of texts, as well as for working with ambiguities in the Ukrainian language.

To solve NLP problems for the Ukrainian language, you can use Stanza [22]. Stanza is a Python package for tokenization, lemmatization, POS tagging, dependency analysis, and NER problem-solving.

There is also another Python package - NLP-Cube [23], the purpose of which is to solve the problems of tokenization, splitting sentences into parts, multi-word tokenization, lemmatization, tagging parts of speech, and dependency analysis.

In [24] presents a morphological analyzer for the Russian and Ukrainian languages - pymorphy2. Its functionality includes POS tagging, a word declension engine, and lemmatization.

In [25], a stemmer for processing the natural Ukrainian language - Tree_stam - is provided. It is based on machine learning and is used to reduce a word to its basic form. For example, for the words "running", "runner", and "run", the stemmer can return the same stem "run". This helps reduce the number of unique words in the text and simplify further analysis. Tree_stam outperforms all stemmers available today, as well as some lemmatizers. It also has the lowest percentage of underestimation errors compared to existing recognition algorithms.

Pre-trained machine learning models for the Ukrainian language

Practice shows that using pre-trained models has several advantages compared to creating models "from scratch", namely:

- Efficient use of resources: Pre-trained models already have a large number of parameters that have been optimized during training on large volumes of data.
- General knowledge of the data: This allows them to make better generalizations and more accurate predictions.
- Less need for training data.

- Reduction of training time, as it will only be necessary to adjust the model for specific data (fine-tuning process).

It is important to understand that when choosing a model for fine-tuning, you should take into account the problem being solved. The model has been previously trained on a similar or related problem and with a similar architecture. For GEC models for the Ukrainian language, you need to choose models that are trained specifically on Ukrainian data. Algorithms of GEC models in other languages will not always be suitable, since the Ukrainian language is morphologically complex and therefore often requires separate approaches.

Since the GEC task refers to text processing, the architecture should be chosen as sequence2tagging, NMT, or with the Transformer architecture.

Below is a list of pre-trained models that can be used to solve GEC problems in the Ukrainian language:

- M2M-100 – machine translation of Ukrainian from/to 100 languages.
- Sequence-to-sequence models
- mBART50 is an advanced model of mBART and is designed for multilingual translation.
- mT5 – multilingual translation model.
- aya-101 – a model that supports 101 languages. It has 13 billion parameters.
- pythia-uk – updated mT5 on wiki and oasst1 for chats in Ukrainian.
- UAlpaca – improved Llama fine-tuned to execute machine translation instructions on the Alpaca dataset.
- XGLM – autoregressive model. It has 4.5 billion parameters.
- Tereveni-AI/GPT-2
- uk4b and haloop inference toolkit – GPT-2 models of small, medium, and large sizes, trained on Wikipedia, news, and UberText 2.0 books.
- xlm-roberta-base-uk – a shortened version of XLM-RoBERTa for the Ukrainian language.
- youscan/ukr-roberta-base – a model of the Ukrainian language that solves the task of finding a masked word - fill_mask.
- Electra is one of the newest models in the field of text representation, which is characterized by high efficiency and accuracy. The main idea behind the Electra model is that it uses two models that interact with each other: a generator and a discriminator. The generator creates noisy examples by replacing some tokens of the input text with artificially generated tokens. The discriminator tries to

distinguish the original examples from those generated by the generator. Electra uses a more efficient pre-training method compared to BERT. Instead of training a model on the token masking task, Electra trains a generator to artificially create examples. Electra can achieve high accuracy even with fewer parameters compared to other architectures such as BERT or GPT.

- Helsinki-NLP – machine translation of Ukrainian from/to 25 languages.
- MITIE NER Model, ukr-models/uk-ner, lang-uk/flair-uk-ner, dchaplinsky/uk_ner_web_trf_large — models for solving NER problems.
- lang-uk/flair-uk-pos – a model for solving POS tagging problems
- uk-punctcase – punctuation and case recovery model based on XLM-RoBERTa-Uk.
- punctuation_uk_bert – Another punctuation and case recovery model based on bert-base-multilingual-cased.
- ukrainian-word-stress – adds stress to words.

Application of machine learning models and corpora for the Ukrainian language

Most of the GEC systems for English that show the best results in the tests are based on the NMT architecture. In [26], the Pravopisets team conducts research on the use of data augmentation, that is, the use of synthetically generated data for training the GEC model. As a basis, mBart-50 was chosen as the NMT model, XLM-ROBERTa as the seq2tag model, as well as the UA_GEC corpus, on which training will be carried out. For seq2tag, tokens were tagged with KEEP, DELETE, APPEND, or REPLACE tags, depending on what to do with them. The classification algorithm was borrowed from the GECToR team.

The MBart-50-large model was configured with the task of translation from Ukrainian to Ukrainian, fine-tuned on the UA-GEC corpus with augmentation by a synthetically generated data set. 5,000 sentences were generated using circular transliteration (ukr-rus-ukr), 10,000 sentences using the generation of punctuation errors, 2,000 blurring of the text, and 10,000 Russianisms were added to this model.

Below you can see examples of the work of the GEC model of the Spellwriter team:

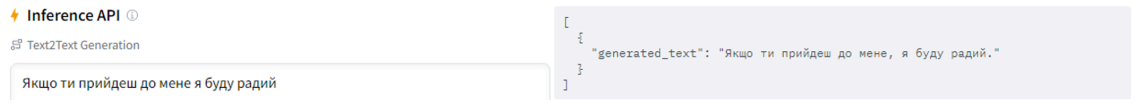

Fig. 5. *An example of successful correction of punctuation errors by the Speller team model.*

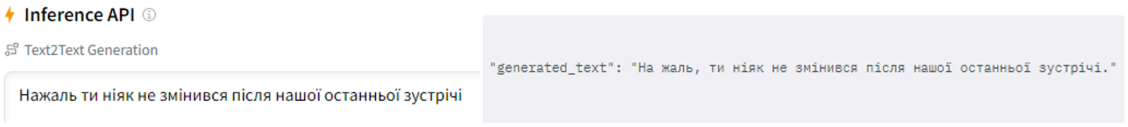

Fig. 6. *An example of successful correction of spelling and punctuation errors*

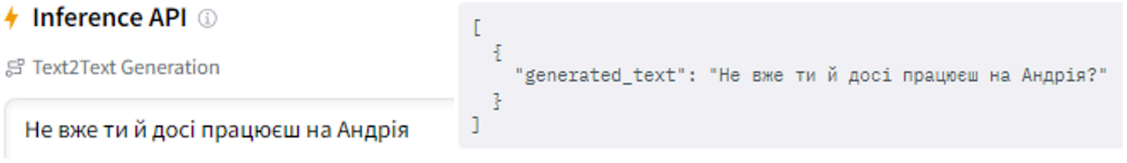

Fig. 7. *An example of correcting not all errors in the text.*

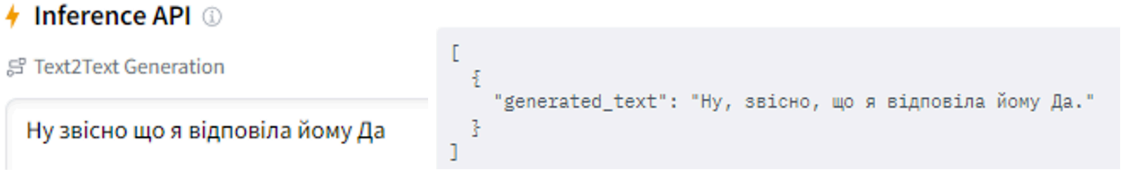

Fig. 8. *An example of not being able to correct Russianism in the text.*

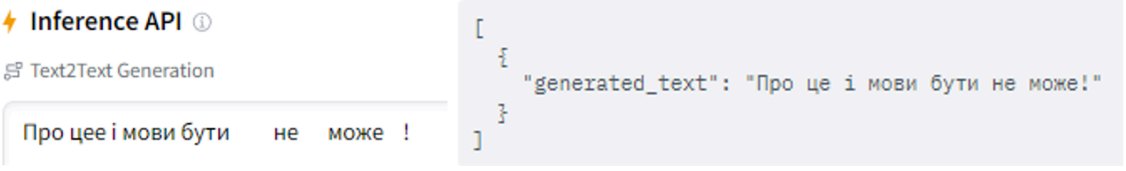

Fig. 9. *An example of successful text formatting.*

doi.org/10.32388/N4VGBJ.2

As a result, the NMT-based model performed better than the rule-based model and the seq2tag model in the evaluation test (Fig. 10).

Type	TP	FP	FN	Total P	Total R	Total F0.5
NMT (baseline)	685	302	1068	0.694	0.391	0.601
NMT (best)	691	241	1047	0.741	0.398	0.632
seq2tag (baseline)	399	753	953	0.346	0.295	0.335
seq2tag (most data)	461	1324	901	0.258	0.339	0.271
rule-based	104	1064	1194	0.089	0.080	0.087

Fig. 10. *Comparison of indicators between different models of the Speller team* [26].

After researching most of the state-of-the-art GEC approaches in the English language and trying to adapt them to the Ukrainian language, it was found that the most effective GEC system can be developed using the NMT approach, but seq2tag has many research opportunities. The best model scored 0.632 for F0.5 on the UA-GEC dataset. Moreover, the results show that adding synthetic data to the UA-GEC training data gives the best results. It becomes clear that the quality of the data corpus is much more important than the size of this set. Therefore, the authors [26] concluded that the most promising direction for future research is the use of human-annotated GEC data.

GrammarUA (smartik/mbart-large-50-finetuned-gec) uses the improved mBART50, which also showed good results for languages with limited resources on the joint task [27] (Fig. 11).

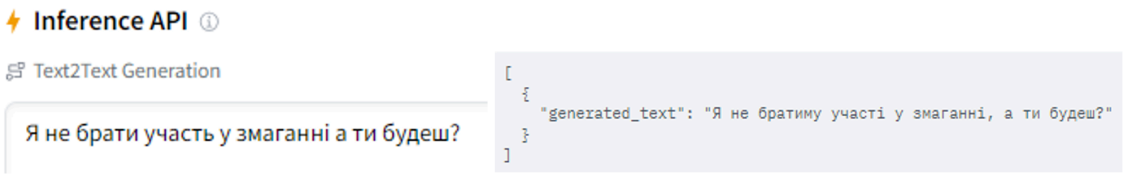

Fig. 11. *An example of implementation of the GrammarUA model.*

The schhwmn/mt5-base-finetuned-ukr-gec model was created and trained mT5 model on the UA-GEC training corpus. The peculiarity of this model is that the author did not use the entire UA-GEC corpus, but only the part where there are errors. An example of the operation of this model can be seen below:

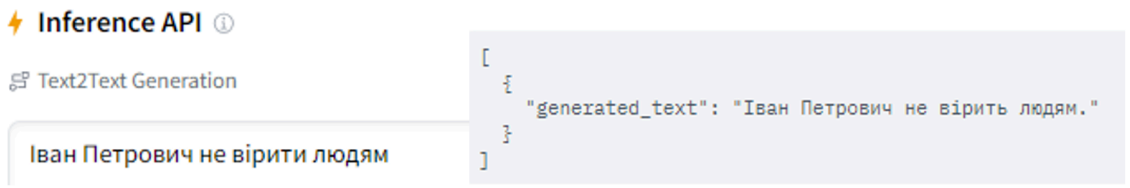

Fig. 12. *An example of successful execution of the model.*

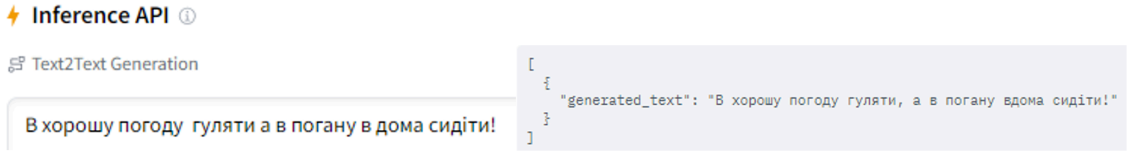

Fig. 13. *An example of incorrect text correction.*

WebSpellChecker [28] used a proprietary transformer architecture called RedPenNet. The architecture uses a pre-trained MLM encoder together with a shallow decoder to generate replacement tokens and gaps for editing GEC instances. During the generation of edit markers, the encoder and decoder attention weights determine the edit ranges (start and end) that indicate the position of the edit in the original sentence. Edit tokens are predicted by autoregressive method.

The main advantage of RedPenNet is the ability to implement any source-to-target transformation with a minimal number of autoregression steps, which enables efficient handling of GEC cases, including correlated and multipoint edits. However, due to its specific architecture, RedPenNet is not suitable for training or fine-tuning. Thus, convenient tools such as the HuggingFace infrastructure cannot be used to quickly set up and deploy the model.

The best model for solving GEC problems for the Ukrainian language is the QC-NLP team model [29], which was presented at a joint task [27]. The team tuned a large multilingual model (mT5) in two stages: first on synthetic data and then on UA_GEC data. They also used a smaller seq2seq Transformer model pre-trained on synthetic data and refined on the UA_GEC corpus. A key innovation pioneered by the QC-NLP team is step-by-step tuning, first on synthetic and then on "golden" data.

The model indicators are shown in the figure below.

		GEC+Fluency			GEC only		
Model	**Number of params.**	**P**	**R**	**$F_{0.5}$**	**P**	**R**	**$F_{0.5}$**
mT5 large	1.3B	73.21	53.22	68.09	76.81	61.39	73.14
seq2seq	275M	69.91	53.78	65.96	72.32	63.13	70.27

Fig. 14. *Indicators of the GEC model of the QC-NLP team* [29].

Conclusions

For now, existing GEC tools and systems for the Ukrainian language are not always able to detect all grammatical errors or provide accurate corrections. They are still weak in declension and identifying ambiguities, although they are quite good at correcting punctuation errors.

Identifying grammatical errors often requires understanding the context. GEC systems often do not correctly interpret the meaning of a sentence and therefore do not identify and correct errors accordingly.

Further development of neural network architectures may improve the quality of correction. Using more powerful and deeper models can help improve the situation. Models should take better account of syntactic and semantic relationships between words. Currently, all machine learning models use pre-trained models that were not created taking into account the morphological complexity of the Ukrainian language, which is why the improvement of GEC systems directly depends on an increasing number of studies aimed at studying the peculiarities of the Ukrainian language. Taking into account declension, word formation, and other features of the language is important.

To sum up, it is necessary to concentrate attention and efforts on creating a specialized model that will take into account all the nuances of the Ukrainian language. Moreover, it is needed to provide a much larger "gold" standard hand case exclusively for GEC tasks. Finally, it is necessary to continue research with the training of models on synthetically created data sets.

Statements and Declarations

Author Contributions

Conceptualization: R.F., V.V.; Methodology: R.F.; Investigation: R.F.; Resources: R.F.; Writing – Original Draft: R.F.; Writing – Review & Editing: V.V.; Visualization: R.F.; Supervision: V.V.; Project Administration:

V.V.

Data Availability

Data sharing is not applicable to this article as no new data were created or analyzed in this study. All sources analyzed are included in the reference list.

References

1. ^*Bryant C, Yuan Z, Qorib MR, Cao H, Ng HT, Briscoe T (2022). "Grammatical error correction: A survey of th e state of the art." CoRR. abs/2211.05166.*
2. ^*Syvokon O, Nahorna O (2021). "UA-GEC: grammatical error correction and fluency corpus for the ukrainia n language." CoRR. abs/2103.16997.*
3. ^*Smith OB, Ilori JO, Onesirosan P (1984). "The proximate composition and nutritive value of the winged bea n Psophocarpus tetragonolobus (L.) DC for broilers." Anim. Feed Sci. Technol. **11**:231–237.*
4. a, b*Naghshnejad M, Joshi T, Nair VN (2020). "Recent Trends in the Use of Deep Learning Models for Gramm ar Error Handling." arXiv. 2009.02358.*
5. ^*Brockett C, Dolan WB, Gamon M (2006). "Correcting ESL Errors Using Phrasal SMT Techniques." In Procee dings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the A ssociation for Computational Linguistics. Sydney, Australia: Association for Computational Linguistics. 249 –256.*
6. ^*Yoshimoto I, Kose T, Mitsuzawa K, Sakaguchi K, Mizumoto T, Hayashibe Y, Komachi M, Matsumoto Y (201 3). "NAIST at 2013 CoNLL grammatical error correction shared task." In Proceedings of the Seventeenth Con ference on Computational Natural Language Learning: Shared Task. Sofia, Bulgaria: Association for Comp utational Linguistics. 26–33.*
7. ^*Junczys-Dowmunt M, Grundkiewicz R (2014). "The AMU system in the CoNLL-2014 shared task: Grammat ical error correction by data-intensive and feature-rich statistical machine translation." In Proceedings of t he Eighteenth Conference on Computational Natural Language Learning: Shared Task. Baltimore, Marylan d: Association for Computational Linguistics. 25–33.*
8. ^*Felice M, Yuan Z, Andersen E, Yannakoudakis H, Kochmar E (2014). "Grammatical error correction using h ybrid systems and type filtering." In Proceedings of the Eighteenth Conference on Computational Natural L anguage Learning: Shared Task. Baltimore, Maryland: Association for Computational Linguistics. 15–24.*

9. ^a, ^b *Cho K, van Merrienboer B, Gülçehre C, Bougares F, Schwenk H, Bengio Y (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation."*
10. ^ *Yuan Z (2017). "Grammatical error correction in nonnative English." University of Cambridge, Computer Laboratory. Tech. Rep.*
11. ^ *Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017). "Attention is all you need."*
12. ^ *Wolf T, Debut L, Sanh V, Chaumond J, Delangue C, Moi A, Cistac P, Rault T, Louf R, Funtowicz M, et al. (2020). "Transformers: State-of-the-Art Natural Language Processing." In Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations. ACL Anthology. Online. 38–45.*
13. ^ *Chaplynskyi D (2023). "Introducing UberText 2.0: A Corpus of Modern Ukrainian at Scale." In Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP). Dubrovnik, Croatia: Association for Computational Linguistics. 1–10.*
14. ^ *Starko V, Rysin A, Havura O, Cheilytko N, et al. (2016-2023). "BRUK: Braunskyi korpus ukrainskoi movy." https://github.com/brown-uk/corpus.*
15. ^ *Shvedova M, von Waldenfels R, Yarygin S, Rysin A, Starko V, Nikolajenko T, et al. (2017-2022). "GRAC: General regionally annotated corpus of Ukrainian."*
16. ^ *Abadji J, Suarez PO, Romary L, Sagot B (2022). "Towards a cleaner document-oriented multilingual crawled corpus." In Proceedings of the Thirteenth Language Resources and Evaluation Conference. Marseille, France: European Language Resources Association. 4344–4355.*
17. ^ *Kotsyba N, Moskalevskyi B, Romanenko M, et al. (2018). "Laboratorija ukrajins'koji." https://mova.institute/.*
18. ^ *Darchuk N (2017). "Possibilities of semantic markup of the Ukrainian language corpus (KUM)." Naukovyi chasopys Natsionalnoho pedahohichnoho universytetu im. M.P. Drahomanova. abs/1911.02116:18–28.*
19. ^a, ^b, ^c *Syvokon O, Nahorna O, Kuchmiichuk P, Osidach N (2023). "UA-GEC: Grammatical Error Correction and Fluency Corpus for the Ukrainian Language." In Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP). Dubrovnik, Croatia: Association for Computational Linguistics. 96–102.*
20. ^ *LanguageTool. About. https://languagetool.org/about.*
21. ^ *LanguageTool API NLP UK. https://github.com/brown-uk/nlp_uk.*
22. ^ *Stanza – A Python NLP Package for Many Human Languages. https://stanfordnlp.github.io/stanza/.*
23. ^ *NLP-Cube. https://github.com/adobe/NLP-Cube.*

24. ^Pymorphy. https://github.com/pymorphy2/pymorphy2.

25. ^Tree_stem - Stemmer for the Ukrainian language. https://github.com/amakukha/stemmers_ukrainian.

26. a, b, cBondarenko M, Yushko A, Shportko A, Fedorych A (2023). "Comparative Study of Models Trained on Synthetic Data for Ukrainian Grammatical Error Correction." In Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP). Dubrovnik, Croatia: Association for Computational Linguistics. 103–113.

27. a, bThe UNLP 2023 Shared Task on Grammatical Error Correction for Ukrainian. https://aclanthology.org/2023.unlp-1.pdf.

28. ^Didenko B, Sameliuk A (2023). "RedPenNet for grammatical error correction: Outputs to tokens, attentions to spans." In Proceedings of the Second Ukrainian Natural Language Processing Workshop. Dubrovnik, Croatia: Association for Computational Linguistics.

29. a, bGomez FP, Rozovskaya A, Roth D (2023). "A low-resource approach to the grammatical error correction of Ukrainian." In Proceedings of the Second Ukrainian Natural Language Processing Workshop. Dubrovnik, Croatia: Association for Computational Linguistics.

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.