Mathematical model of a decision support system for identification and correction of errors in Ukrainian texts based on machine learning

Rostyslav Fedchuk*,† and Victoria Vysotska†

Information Systems and Networks Department, Lviv Polytechnic National University, 12 Bandera Str., 79013 Lviv, Ukraine

Abstract

This research presents a mathematical model for a decision support system aimed at identifying and correcting errors in Ukrainian-language texts. The system combines two key components: error detection as token-level multi-class classification and error correction as context-aware text generation. Special attention is given to the structural and grammatical complexity of the Ukrainian language. Probabilistic models and machine learning techniques are used to classify errors and suggest corrections. The model accounts for dependencies between tokens and linguistic features specific to Ukrainian. Methods for text vectorization are mathematically justified based on morphological and syntactic characteristics. The correction module generates accurate replacements for erroneous tokens using contextual modeling. This approach enables high accuracy and adaptability in processing Ukrainian texts. The proposed system provides a universal foundation for automated Ukrainian text editing.

Keywords

Error identification, error correction, NLP, GEC, Ukrainian language, machine learning, text generation, probabilistic model, text processing, morphology.

1. Introduction

In today's rapidly growing text content, the problem of automatic text verification and correction is becoming particularly urgent. For the English language, many studies have been conducted, and significant progress has been made in solving the GEC problem [1]. However, for the Ukrainian language, there is still a lack of sufficient qualitative research that would consider the peculiarities of the Ukrainian language, its morphological structure, dialect diversity, syntax multifacetedness, and context of dependencies. Errors in texts can be of different nature – spelling, grammatical, punctuation, semantic, etc., and their correction becomes necessary to improve the quality of information, as well as to ensure its correct perception by users.

The main approach to solving the GEC problem includes two interrelated stages: identification of errors in the text and generation of corrections. Historically, traditional methods were used to solve these problems, but their effectiveness is significantly inferior to modern machine learning models.

Traditional GEC systems are typically built on a set of linguistic rules [2] formulated by experts and dictionary databases. Such systems check spelling, number, gender, or case agreement between words, and even basic punctuation. For example, they are excellent at detecting erroneous word forms ("великий книга") or separate spelling of particles ("на приклад" → "наприклад"). However, the complexity and multifacetedness of the grammatical dependencies of the Ukrainian language impose numerous limitations on such systems. They do not consider contexts in which rules often change their interpretation, for example, in colloquial or artistic language, where verbal abbreviations, non-standard constructions, or stylistic variations occur. In addition, creating and

*MoDaST 2025: Modern Data Science Technologies Doctoral Consortium, June, 15, 2025, Lviv, Ukraine

1* Corresponding author.

† These authors contributed equally.

rostyslav.b.fedchuk@lpnu.ua (R. Fedchuk); victoria.a.vysotska@lpnu.ua (V. Vysotska);

0009-0002-6669-0369 (R. Fedchuk); 0000-0001-6417-3689 (V. Vysotska);

maintaining an extensive set of rules for such systems requires significant human and time resources. As a result, traditional systems demonstrate low performance in real-world conditions and have hardly adapted to new data and text styles.

More sophisticated systems based on statistical machine translation (SMT) were the next step in the development of the GEC problem and offered a more flexible approach to text correction. The basis of such models is the analysis of large corpora of texts, where for each sentence a correct and an incorrect version was known. Using statistical regularities, the models suggest the most likely correction for each erroneous fragment. Although this approach has some improvements compared to the previous one, it remained limited due to its dependence on the quality of the training corpus. In addition, SMT models do not sufficiently consider the global context of the text, which leads to many stylistic and grammatical errors in complex constructions. The modern era of solving the GEC problem includes the use of machine learning, which has significant advantages compared to traditional approaches. Machine learning models, such as transformative architectures, can consider not only local, but also global dependencies between words in the text. They provide context analysis, which is critical for classification and generation tasks within the GEC problem. A classification approach is used to identify errors, where each token of the text is analyzed and labeled with a corresponding label. In the Ukrainian language, this stage faces significant difficulties associated with morphological complexity, since most words can take on different forms depending on their case, number, or gender. Also, grammatical dependencies are often determined by the interaction of several tokens located at a considerable distance from each other in the text, such as when agreeing the subject and predicate in a complex sentence. The complexity of error classification is compounded by the polysemy of lexemes, where the meaning of a word changes depending on the semantic context.

Error correction, on the other hand, is a text generation task where the system has to predict the correct correction for each erroneous token or fragment of text. Here, the key is not only the accuracy of grammar, but also the coordination of the corrected text with a style that matches the context of the input text. Generative models, such as mT5 [3], have proven their high efficiency in solving this problem due to the contextual analysis of each token and the ability to work with multi-level dependencies. For the Ukrainian language, this is especially important due to the significant semantic load of dependencies between words and syntactic variations, which can complicate the generation of corrections even for standard grammatical constructions. The use of machine learning methods has significant advantages. Models built on such architectures are adaptive in working with real texts and can qualitatively consider the context. They are devoid of the strict limitations inherent in linguistic rules and can be trained on large data corpora, gradually improving their predictions. Transformer models, such as BERT for identification and T5 for correction, allow combining both stages within a single system that can work with complex error cases.

The purpose of the research is to develop a mathematical model of a decision support system for identifying and correcting errors in Ukrainian-language texts using machine learning methods. The main tasks are to formalize the processes of text analysis, build a model that considers the morphological, syntactic and contextual specifics of the Ukrainian language, build DFD diagrams, as well as create algorithms for detecting errors and correcting them.

The object of the research is the processes of automatic identification and correction of errors in Ukrainian texts, considering their morphological and syntactic complexity. The subject of the research is a mathematical model and machine learning algorithms that allow solving classification and generation problems for building an effective text processing system.

2. Statement of the problem

The problem of correcting errors in a text belongs to the category of GEC tasks, which include two main subtasks: error identification and error correction.

Today, there are several reasons that complicate the solution of this problem for Ukrainian-language texts. First, most traditional automatic text checking systems are designed for less complex languages from a grammatical point of view, such as English. The application of these systems to the Ukrainian language turns out to be ineffective, since they do not consider the specifics of its grammar and syntax. For example, the variability of word forms (cases, numbers, genders, declension) complicates the task of identifying erroneous tokens. In addition, the Ukrainian language has significant lexical-semantic dependencies that depend on the context.

Another challenge is the variety of errors that can occur in texts. Spelling errors are relatively straightforward to process automatically, while grammatical, punctuation, and even stylistic errors require detailed analysis of the context and interactions between words within a sentence or even a paragraph. For example, in the sentence "Мама купив хліб," the error ("купив" instead of "купила") requires modeling the agreement between the subject and the predicate. Identifying such complex dependencies and creating mechanisms to correct them remains an open task.

In addition, the task is complicated by the lack of large open corpora of training data for the Ukrainian language that would provide sufficient quality for training machine learning models. The dominance of English-language content in the field of NLP creates an uneven distribution of resources and methodologies, which requires adapting existing technologies to the specifics of the Ukrainian language.

The GEC task is based on two key stages, which are performed sequentially: error identification and correction. The error identification stage is responsible for identifying erroneous tokens in the text and the type of error they contain. The work process at this stage includes the analysis of each token and an assessment of the probability of an error, considering the context of the surrounding words.

To successfully solve the GEC problem, a system is needed that is able to integrate the identification and correction processes into a single model-oriented approach and solve the following challenges of the Ukrainian language: contextual ambiguity, morphological distribution (it is necessary to take into account 7 cases of Ukrainian words, their number, gender, form), syntactic complexity, and the lack of training corpora in the Ukrainian language.

3. Related works

Modern automatic text correction tools have become an important tool in overcoming various types of spelling, grammar, punctuation and stylistic errors. In this context, the most well-known and effective solutions are software applications and platforms integrated into word processors, online tools and machine learning systems. Grammarly [4] is one of the most popular tools for automatic text checking. Unfortunately, Grammarly does not currently support the Ukrainian language for checking grammar, spelling or stylistics. The main functionality of the service is focused on the English language. However, the company is actively working on the development of the Ukrainian language in the field of computational linguistics. Grammarly created [5] and published in the open access annotated GEC corpus of the Ukrainian language [6], which contains almost 34,000 sentences. This resource is intended for scientific and practical study of the language, as well as for training and evaluation of grammar correction programs.

GPTTools.ai [7] is a Ukrainian desktop application based on GPT-4 artificial intelligence, designed to effectively work with texts in Ukrainian and more than 70 other languages. The tool provides the ability to generate texts, edit, correct stylistic, spelling and grammatical errors, as well as translate. A key feature is support for individual prompts – users can create their own templates to automate routine tasks. The program is suitable for students, scientists, copywriters and anyone who works with large amounts of text information, ensuring convenience and accuracy of data processing. LanguageTool [8], which supports a multilingual format, including Ukrainian, is currently one of the most functional tools for text verification. LanguageTool works based on rules and currently has 1219 rules [9]. This platform is available as an online tool, browser extension or desktop application. It can work with grammatical, spelling and contextual errors, offering recommendations for their correction. LanguageTool integration is possible through the LanguageTool API, which supports the Ukrainian language and is called NLP-uk [10]. Through this integration, the user gets access to normalization, tokenization, lemmatization, POS-tagging of texts, as well as tools for solving the problem of ambiguities in the Ukrainian language.

The word processors Microsoft Word and Google Docs also have a text proofing function, including basic support for the Ukrainian language. Microsoft Word allows you to correct spelling errors using dictionary databases and linguistic rules. Google Docs offers a similar basic proofing, but its functionality is less developed. For complex text analysis, these platforms can use additional integrations, such as LanguageTool.

Among the specialized Ukrainian tools, OnlineCorrector [11] should be highlighted, which is focused on checking spelling, punctuation, and style in texts. This online tool allows you to upload texts for checking and receive a result with corrected errors. It pays special attention to working with texts of various formats, considering the specifics of the Ukrainian language.

Modern machine learning models, particularly GECToR [12], open up new opportunities for automating text correction processes. This model is built on a transformative architecture and is available for adaptation to any language, including Ukrainian. GECToR uses an attention mechanism to consider the context in text correction, which allows achieving high accuracy and efficiency.

Today, platforms such as Hugging Face [13] are also available to developers, which offer pre-trained models, including multilingual versions of BART, BERT, GPT, or T5. These platforms allow for the creation of customized solutions for GEC and other natural language processing tasks by adapting the models to the specifics of the language and style of the text.

A great example of an interactive multilingual tool is modern GPT models, including ChatGPT, which can not only check and correct texts, but also adapt to complex stylistic features, considering the context of the text. These models can work with any type of text, providing both basic checking and context-based corrections.

In [14], the study examines the effectiveness of using GPT-3.5 for grammatical error correction (GEC) in a multilingual context in various scenarios, including zero-shot learning, fine-tuning, and using the model to rank correction hypotheses generated by other models. The authors evaluate the performance of GPT-3.5 through automated evaluation methods, such as grammaticality assessment using language models (LM), Scribendi test, and semantic embedding analysis. The results demonstrate that for English, GPT-3.5 shows a high level of consistency in corrections, preserving the semantic integrity of the text, due to which the model corrects errors well and generates smooth sentences. However, in other languages, including Ukrainian, Czech, German, Russian, and Spanish, GPT-3.5 often significantly modifies the source sentences, which sometimes leads to changes in their semantics and creates difficulties for evaluation. The main feature of this work is the detection of GPT-3.5's tendency to overcorrect, as well as the analysis of its limitations in correcting punctuation errors, tense agreement, syntactic dependencies, and lexical compatibility. Despite the powerful capabilities of the model, the authors emphasize the problems of its adaptation for GEC tasks in a multilingual environment, especially for languages with a rich grammatical structure, such as Ukrainian.

In [15], the authors proposed a new approach to multilingual grammatical error correction (GEC), which is based on the use of pre-trained machine translation models. A feature of their approach is the integration of neural machine translation methods into the GEC task, which made it possible to effectively adapt the models to work with texts in different languages, in particular low-resource ones. The authors managed to overcome the limitations of traditional GEC models by optimizing pre-trained transformers, which ensured high accuracy of correction, context-awareness, and naturalness of the corrected text. The main difference of their work is that they adapted translation models, in particular transformers, for multilingual correction, while maintaining their universality and ability to scale to new language sets. This allows not only to correct errors with high accuracy, but also to effectively work with languages for which limited training resources are available.

4. Mathematical model of a decision support system

The decision support system for identifying and correcting errors in Ukrainian-language texts is based on the use of natural language processing (NLP) and machine learning methods [16-24]. The main goal of the DSS is to automatically identify errors in the text, correct them or provide recommendations for correction. The system consists of three main modules: an error identification module, an error correction module, and a machine learning module. This modular structure allows you to easily update the system, adapt it to different tasks, and increase the accuracy of error recognition and correction. An important aspect is the ability to integrate additional components, such as lexical databases, semantic networks, and algorithms for increasing the accuracy of text analysis.

The error identification module performs text analysis to detect incorrect words, grammatical, spelling, and stylistic errors. Its main tasks are:

- Text segmentation – dividing the input text into individual sentences.
- Stop word removal – removal of punctuation marks, as well as other language constructs that do not carry a semantic load and will only slow down the identification of errors.
- Text tokenization – splitting the input text into individual words or phrases for further analysis.
- Lemmatization, stemming and morphological analysis – determining the basic form of words, their part of speech and grammatical characteristics.
- Classification of words into correct and potentially incorrect – application of machine learning methods to assess the correctness of words in the context of a sentence.
- Contextual analysis – assessment of the probability of an error based on surrounding words (n-gram models, transformers, rules).

Based on these stages, a list of potential errors is formed, which is transferred to the correction module. After identifying errors, the system should offer appropriate corrections. The main functions of the error correction module are as follows: generation of corrections, contextual checking, ranking of correction options, calculation of the model's degree of confidence in the correction.

Depending on the settings, the system can either automatically make corrections or prompt the user to choose the most appropriate option. In some cases, the option of interactive training by the user is possible.

The machine learning module is critical for system adaptation and improvement. Its main functions include model training, parameter updating, and feedback collection.

In the process of building a mathematical model of a decision support system (DSS) for identifying and correcting errors in Ukrainian-language texts, an important stage is the formalization of input and output parameters. This allows you to clearly determine what data the system processes and what results it should generate.

The system receives text in various forms as input, which may contain spelling, grammatical, punctuation, and stylistic errors. The input data is formalized as follows:

$$X = \{x_1, x_2, \ldots, x_n\}, \tag{1}$$

where: X is the input text array consisting of n tokens (words, symbols, sentences); x_i is a separate token or word in the text.

Additionally, auxiliary parameters can be used: text format (plain text, HTML, XML, JSON); language (the system is focused exclusively on the Ukrainian language), contextual metadata (text genre (scientific, journalistic, colloquial), style (formal/informal)).

During the processing process, the text goes through several stages of analysis. The main intermediate parameters are linguistic features (POS tags $P(x_i)$ morphological features $M(x_i)$, syntactic dependencies $S(x_i, x_j)$); probabilistic characteristics (estimated probability that a word contains an error in a given context, confidence model); candidates for corrections, error labels.

The result of the system is the corrected text and correction quality metrics. Formally, the output data can be presented as a set:

$$\hat{X} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\} \tag{2}$$

where: $\hat{X}$ – corrected text, $\hat{x}_1$ – corrected version of the word or token $\hat{x}_i$.

The input parameters include the corrected text, a list of errors found, and correction evaluation metrics, including accuracy, completeness, F1-measure, and confidence score.

In the process of developing a decision support system for identifying and correcting errors in Ukrainian-language texts, it is important to consider several limitations that affect its functionality, as well as clearly define the performance criteria that allow assessing the quality of its work. These aspects are of decisive importance for building a reliable and accurate mathematical model.

One of the key limitations is the linguistic specificity of the texts with which the system works. The Ukrainian language has a rich morphological structure, a significant number of grammatical rules, and numerous exceptions, which complicates the process of automated analysis. An important challenge is the processing of polysemy – cases when one word can have several meanings depending on the context. In addition, the system may encounter difficulties when processing neologisms, professional vocabulary, or dialect features that are not always included in the training corpora.

Technological constraints also play an important role in the system's performance. One critical aspect is the performance of the algorithms used to analyze and correct text. Some modern natural language processing models, such as neural network transformers, require significant computing resources, which can create problems when used in real time or on devices with limited computing power. In addition, the length of the text being analyzed can impose limitations on the system's performance, since large fragments must be divided into smaller parts, which can reduce the accuracy of contextual analysis.

Defining performance criteria is an important stage in assessing the quality of the system. The main indicator is the accuracy of error detection and correction, which is evaluated using metrics such as precision, recall, and F1-measure. High precision means that the system detects only those errors that are present in the text, while high completeness indicates the ability to recognize as many errors as possible without omissions. The optimal option is a balance between these indicators, since an excessive number of detected errors without sufficient accuracy can lead to erroneous corrections. Formally, the correction criteria can be defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{3}$$

$$Quality = \frac{TP}{TP + FP + FN}, \tag{4}$$

$$ErrorRate = \frac{FP}{TP + FP}, \tag{5}$$

$$Recall = \frac{TP}{TP + FN}, \tag{6}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}, \tag{7}$$

where TP is the number of correctly identified errors, TN is the number of correct corrections, FP is the number of incorrectly identified errors, FN is the number of missed errors.

In addition to accuracy, an important criterion is the speed of the system. Text processing time should be minimal, especially in interactive applications where the user expects quick feedback. Performance can be measured by estimating the number of words or sentences that the system is able to process per second. Optimization of computational algorithms allows to reduce the load on the processor and memory, which is especially important for integrating the model into mobile applications or web services.

$$T_{avg}=\frac{1}{N}\sum_{i=1}^{N} T_i, \tag{8}$$

where T_i is the processing time of the i-th sentence, N is the total number of sentences in the test corpus.

Another important aspect is the resource consumption, which can be estimated using random access memory (RAM) and processor time (CPU). For example, the average resource usage is determined by the formula:

$$R_{avg}=\frac{1}{N}\sum_{i=1}^{N} R_i, \tag{9}$$

where R_i is the amount of memory used during processing the i-th text fragment.

Another important aspect is the quality of the corrected text. Even if the system correctly identifies errors, its corrections must be relevant and correspond to the style and semantics of the original text. This parameter is assessed using metrics that compare the corrected text with the reference text, such as BLEU [17] or METEOR, as well as through manual assessment by experts or feedback from users. Corrections should not violate the logic or content of the text, which is especially critical for automated correction systems in professional areas, for example, in scientific articles or legal documents.

5. Mathematical model of the error identification in Ukrainian texts

Error identification in texts is one of the fundamental tasks of natural language processing (NLP), which is of particular importance for languages such as Ukrainian, which are characterized by high morphological complexity. The identification task consists in identifying fragments of text that may contain spelling, grammatical, lexical-semantic or syntactic errors, as well as in determining the type of these errors. This is a key stage in automatic text correction systems, where the effectiveness of correction directly depends on the accuracy and quality of identification.

Formally, the task can be formulated as follows. Let X = $\{x_1, x_2, \dots, x_N\}$ be the input text presented as a sequence of tokens (where a token can be a word, symbol or sentence). The task consists in assigning to each x_i a label y_i, corresponding to a certain class of error from the set of classes (C). A typical set of classes is C = $\{c_0, c_1, \dots, c_K\}$, where (c_0) denotes "no error", and $(c_1, \dots, c_K)$ denote specific types of errors (e.g., spelling error, grammatical error, punctuation error, etc.).

The probability that a token (x_i) belongs to a particular class (c_k), is defined as:

$$P(y_i=c_k|x_i, context)=f_{id}(x_i, context), \tag{10}$$

where f_{id} is a recognition (identification) function that considers both the token x_i and its context. The main task of the identification model is to learn to accurately evaluate the function f_{id} which will be able to correctly determine the type of error or its absence.

The Ukrainian language has several specific features that significantly complicate the effective identification of errors. First, this is morphological complexity, which consists in the richness of inflection forms. For example, adjectives, verbs or nouns can change in gender, number and case, which makes it difficult to detect both spelling and grammatical errors. Second, the contextual ambiguity of Ukrainian words, that is, words with the same spelling can have different meanings or functions depending on the context. For example, the word "пішли" can belong to the past tense verb form or act as a colloquial form to describe movement. Therefore, the identification system should consider not only the local analysis of the token, but also syntactic and semantic connections within the entire sentence.

In addition, the variability of regional dialects, borrowings, as well as the use of colloquial and informal constructions characteristic of the Ukrainian language complicates identification. For example, words like "шо" or abbreviations such as "всьо" are rarely found in formal texts but are actively used in informal speech. The consequence of such linguistic diversity is the need to adapt classification models to datasets that include examples of both standard language and colloquial forms.

In addition, it is important to note that the identification of errors in texts is closely related to the processing of different types of errors. Spelling errors can be easily detected using lexical dictionaries (e.g., checking for unrecognized words), while grammatical or punctuation errors depend on the complex structural dependence between words in a sentence.

Thus, the task of identifying errors in Ukrainian-language text appears as a multi-level problem that requires the use of complex mathematical models that can consider both contextual dependencies between words and features of morphology and syntax. The effectiveness of such models largely depends on the quality of machine learning, which is based on large corpora of texts with clearly annotated errors. Since the models used in this problem are mostly based on machine learning algorithms and deep neural networks, the text must be translated into a form that can be used by algorithms for calculation. To this end, the text is represented through mathematically formalized data that considers both the tokens themselves and their context.

To work with modern deep learning methods, the text must be presented as multidimensional vectors, which are numerical representations of the text. The format of the input data stream is represented as:

$$X=\{w_1,w_2,\ldots,w_N\}, \tag{11}$$

where w_i is the vector representation of the token x_i.

Text vectorization involves converting words or tokens into multidimensional numerical vectors – embeddings, which store semantic or contextual information about the words. Two main approaches to vectorization are used: static and dynamic. In the static vector representation approach, each word in the text has a constant vector that does not depend on its context. Methods such as Word2Vec or FastText represent words based on their occurrence in the text. For example, if the word x_i is converted into a vector with dimension d, then the text matrix looks like this:

$$W=\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}, \tag{12}$$

where $W\in R^{N\times d}$ is a matrix of such representations

For example, the FastText method considers subwords (n-gram structures) to model exceptions such as errors or unfamiliar tokens.

Dynamic contextual vector representations are a more modern approach that consider the context in which a word is used. Popular models such as BERT, XLM-RoBERTa, or mBERT generate such dynamic vectors, where for each token x_i in the text, $w_i=f(x_i,context)$, where f is a transform model that takes into account the context within the entire sentence (or even the text). In this approach, the vector w_i not only conveys the meaning of the word but also considers the dependencies between other tokens. This is critical for the problem of error identification, since some errors can only be identified within a broader context.

To identify errors, it is not enough to analyze individual words or tokens. You also need to consider their position in the sentence and their relationship to neighboring elements. Context is modeled in the following ways: a sequence of tokens is processed by recurrent or transform neural networks that store information about previous and next tokens; the use of special mechanisms, such as the attention mechanism, which allows you to highlight significant parts of the text for a particular token. Mathematically, the context can be represented as a combination of tokens in a window of length m around the analyzed token x_i:

$$context(x_i)=\{x_{i-m},\ldots,x_{i-1},x_{i+1},\ldots,x_{i+m}\} \tag{13}$$

where m is the length of the context window. In transformer architectures, the length of the context can cover the entire sentence or even several sentences. Describing the identification problem in a formalized form, it is possible to structure the input data in the form of two interconnected components: a matrix of word vectors and error labels.

The matrix of word vectors W has the form $W \in R^{(N\times d)}$, where N is the number of tokens in the text, and d is the dimension of the vector representations. Error labels (ground truth), represented in the form of a label vector $Y=\{y_1,y_2,\ldots,y_N\}$, where each $y_i \in C=\{c_0,c_1,\ldots,c_K\}$ corresponds to an error class or its absence.

The formalization of the input data should consider the specifics of the Ukrainian language, including declension, conjugation, agreement by gender and number. Ukrainian texts are characterized by abbreviations ("д/з") and regionalisms, which are also important to consider. It is also necessary to ensure adaptability to various variations of the Ukrainian language, which can be found in spoken and literary texts.

In the process of building a system for identifying errors in Ukrainian-language texts, the central task is to develop a mathematical classification model capable of determining whether a text fragment (token or sentence) contains an error, and if so, classifying it by type. Since error classification is a multi-class classification problem, the mathematical model must determine the probability of each fragment belonging to a certain class c_k.

Identifying errors in text is a complex process that includes several stages of processing. The key task is to find those tokens or segments of text that may potentially contain errors (the so-called candidates for correction), based on formal criteria, language rules and statistical features. The algorithm of the identification system consists of several mandatory stages:

1. Pre-processing of text: Normalization, Tokenization, lemmatization (or stemming). Normalization includes removing unnecessary spaces, tabs, special characters that can be noise; unifying the register to lowercase; expanding abbreviations.
2. Analysis of the syntactic and morphological structure of the text. At this stage, the text is processed to isolate its grammatical and syntactic dependencies. For this, POS-tagging and Dependency Parsing are performed. A syntactic structure is created for the sentence that models the interdependencies between words. This allows you to detect types of syntactic errors, for example, incorrect agreement between the subject and the predicate.
3. Creating a language model for context analysis. After syntactic analysis, the text is segmented into context windows that allow us to consider neighboring tokens when identifying errors. Context windows are defined as subsets of tokens: $context(x_i)=\{x_{i-m},\ldots,x_i,\ldots,x_{i+m}\}$, where m is the width of the window. For example, in the sentence "Мій котик завжди їсть морозиво" for the token "їсть" the context window with width $m=2$ will be: [{"котик", "завжди", "їсть", "морозиво"}]. Context is critically

important, since many errors cannot be detected without considering grammatical and semantic dependencies between words.

4. Selecting candidate tokens for correction. At this stage, the system identifies tokens that may potentially contain errors. This is achieved using a combination of rules, lexical knowledge, and statistical features. Candidate tokens can be words, phrases, or symbols that satisfy one or more criteria that signal a possible error. The definition of such tokens is based on a combined approach that considers both linguistic rules and the results of statistical data analysis:

   - Lexical analysis. A word is considered a candidate if it is not included in the dictionary of standard words of the Ukrainian language (for example, "пирог" → "пиріг"). The morpheme structure is also analyzed: incorrect suffixes or prefixes signal possible errors in the word, such as "зробить" → "зроблять".
   - Grammatical agreement analysis. Incorrect agreement in gender, number, or case between words. For example: "У мене була дві книжки" → "було дві книжки".
   - Punctuation analysis. Violation of the rules for constructing complex sentences or missing/extra punctuation marks. For example: "Прийшов додому і почав читати книгу" → "Прийшов додому, і почав читати книгу".
   - Statistical frequency. If a word or phrase is rarely found (or is absent altogether) in a corpus of standard texts, it becomes a candidate.
   - Contextual inconsistency. Tokens that do not match the context. This can be detected using pre-trained models, such as BERT, which predict whether a given word is typical in each context.

5. Candidate filtering. After the initial selection of potential errors, the system filters candidate tokens to exclude false positive predictions. For example: removing technical terms from the candidates that are not in the dictionary due to their specificity or filtering variants that formally comply with language rules (certain rarely used constructions). This reduces the load on the next stage – the correction module, where only probable errors are processed.

As a result of this algorithm, a subset of tokens that are candidates for correction is determined. They are passed to the next level of analysis system, which can make a specific decision on correction.

To visualize the algorithm, a DFD diagram was used, which is a schematic tool for visualizing data flows in the system. DFD allows you to understand the main data flows in the system and helps to clearly and structurally present the error identification process.

At DFD level 0, the system is represented as a single block that interacts with external entities. It shows that the user enters text into the system. The system returns the output: a list of tokens with error labels and error types.

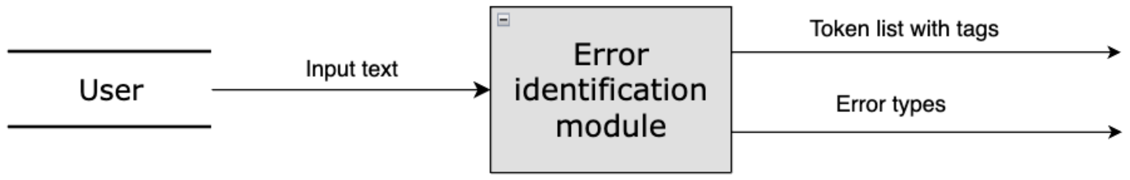


Figure 1: DFD level 0 diagram of the error identification module.

Next, the data flow is detailed, this is the DFD level 1, where the system is divided into main functional blocks.

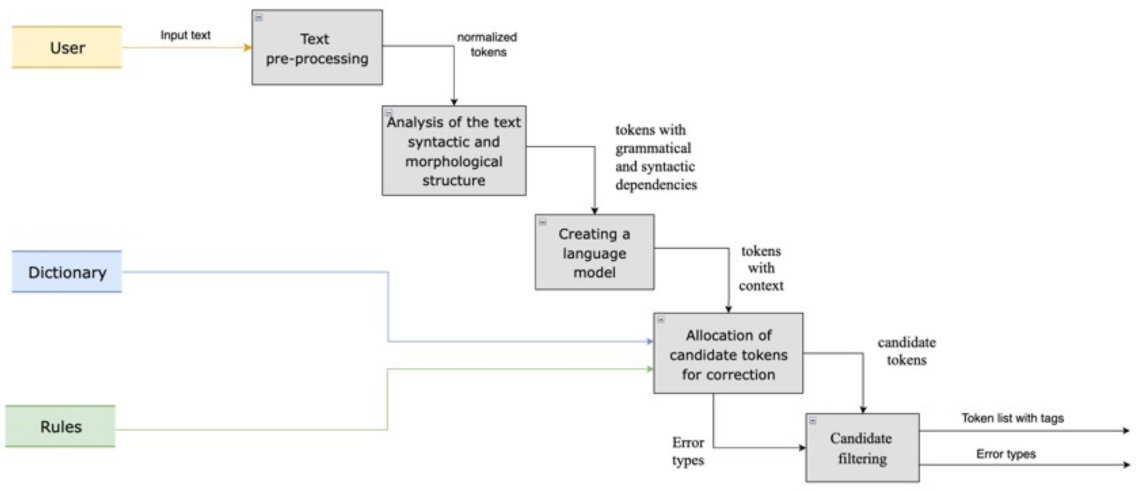

User
Input text
Text pre-processing
normalized tokens
Analysis of the text syntactic and morphological structure
tokens with grammatical and syntactic dependencies
Creating a language model
tokens with context
Dictionary
Allocation of candidate tokens for correction
candidate tokens
Rules
Error types
Candidate filtering
Token list with tags
Error types

Figure 2: DFD level 1 diagram for the error identification module.

DFD level 2 is an extended algorithm. This level displays all the operations that occur within each block.

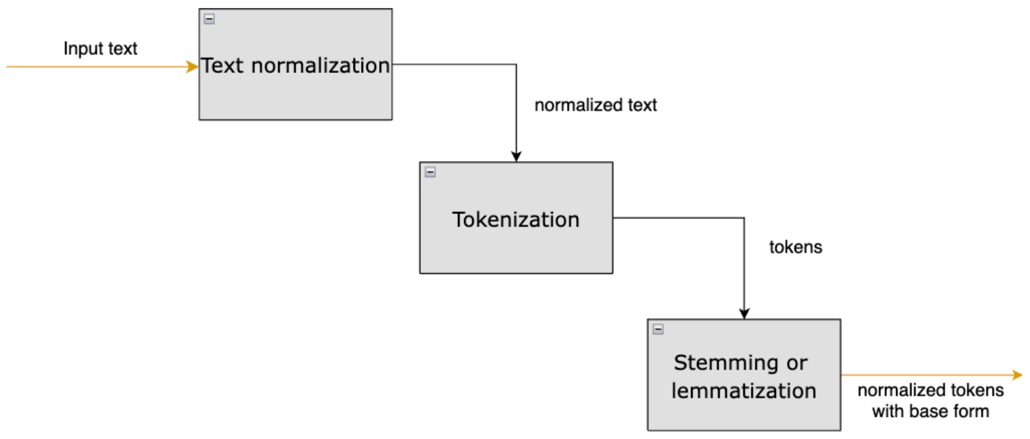


Figure 3: Level 2 DFD diagram of the “Text Preprocessing” block.

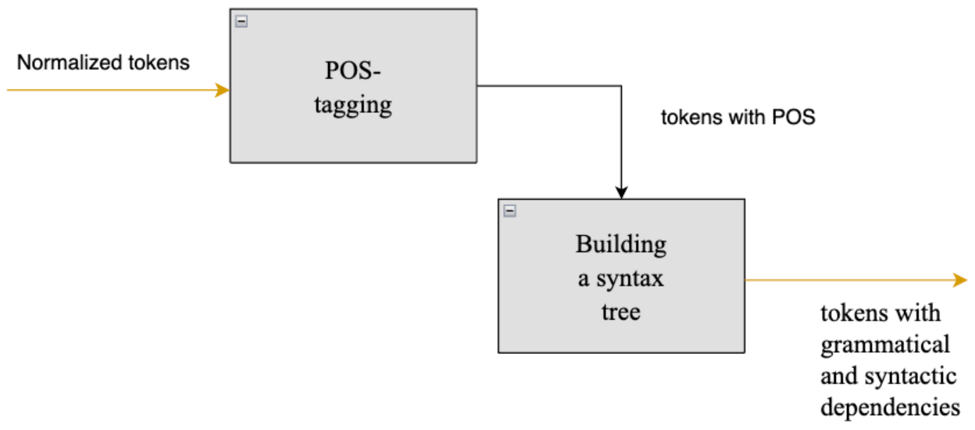


Figure 4: DFD diagram of level 2 of the block “Analysis of the text syntactic and morphological structure”.

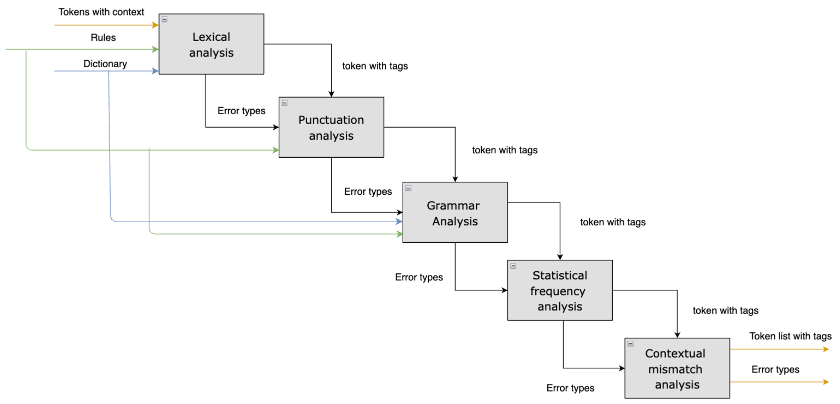


Figure 5: Level 2 DFD diagram of the "Allocation of candidate tokens for correction" block.

6. Mathematical model of the error correction in Ukrainian texts

Error correction in Ukrainian-language texts can be formulated as a text generation task, where the model receives text with errors as input, analyzes the context, determines the correct option instead of the erroneous fragment, and returns the corrected text.

Let the input text be given as a sequence of tokens $X=\{x_1,x_2,\dots,x_N\}$and a sequence of labels $Y=\{y_1,y_2,\dots,y_N\}$, where x_i is a separate token, and $y_i\in C=\{c_0,c_1,\dots,c_K\}$ is the error type for the corresponding token x_i: c_0 - "no error", $\{c_0,c_1,\dots,c_K\}$types of errors (e.g., spelling, grammar, punctuation). The correction task is to transform the text X into the text $\hat{X}=\{\hat{x}_1,\hat{x}_2,\dots,\hat{x}_n\}$, where each corrected token $\hat{x}_i$ is an exact representation without errors, correctly consistent with the context, or unchanged token if there is no error $y_i=c_{0.}$

The main goal of the model is to generate a text $\hat{x}_i$, which complies with the grammatical, stylistic and lexical rules of the Ukrainian language and is consistent with the context of the stored text fragment.

From a mathematical point of view, error correction for each token x_i in the text can be described as the task of finding the optimal candidate $\hat{x}_i$:

$$\hat{x}_i=\underset{x_i'\in D}{arg\,max}\,P(x_i'|x_i,context), \tag{14}$$

where: D is the dictionary set of all valid tokens for the Ukrainian language; $P(x_i'|x_i,context)$ is the conditional probability that the token x_i' is the correct correction of x_i. For each token x_i, the model estimates the probability of different correction options for x_i', given the error and the context around the token. The probability is modeled as follows:

$$P(x_i'|x_i,x_{i-1},x_{i+1},\dots,x_{i-m},x_{i+m})=f_{cor}(x_i,context) \tag{15}$$

where context is all nearby tokens in a window of width 2m, and f_{cor} is the correction function implemented by the text generation model. Error correction requires considering semantic and grammatical context. Semantic context helps to consider consistency with neighboring words, while grammatical context ensures that a word agrees with gender, number, tense, case, etc.

The error correction algorithm works based on input data received from the identification module, which has identified potentially erroneous tokens and classified them by error types. Error correction consists in finding optimal replacements for erroneous tokens consistent with the context, grammatical rules, and stylistic features of the Ukrainian language, and in forming a corrected text.

Description of the error correction algorithm:

1. Initialization of the data structure. This is the initial stage, at which the correction module must receive input tokens X and error labels Y, as well as create an empty list for corrected tokens: $\hat{X}=[\,]$.
2. Processing each token. At this stage, for each x_i in the text X, the error label y_i is checked. If $y_i=c_0$then $\hat{x}_1=x_i$, and the token is added to the list $\hat{X}$ without changes. If $y_i \neq c_0$ (the token is identified as erroneous) then the correction process is initiated according to the error type y_i.
3. Generation of candidates for correction. For each erroneous token $\hat{x}_i$ a set of possible correction options $D=\{d_1, d_2, \ldots, d_M\}$ is determined using the Ukrainian dictionary, grammatical analysis, and a language model. The generation of correct text can be implemented using several approaches: Greedy Search, Beam Search, and Sampling with Temperature. The Greedy Search model sequentially selects the most probable option for each token, but this method does not always guarantee global optimality. During Beam Search, several most probable options are considered, from which the best one is selected based on a global assessment of the context. Sampling with Temperature uses the generation of options adjusted by temperature parameters to increase the variability of corrections. For the problem of generating correct text, the objective function of the correction model optimizes the probability of correct corrected text. For the multilevel evaluation problem, the cross-entropy loss function is used:

$$Loss=-\sum_{i=1}^{N} \log\left(P\left(\hat{x}_i | x_i, context\right)\right), \tag{16}$$

   To identify spelling errors, variants are generated that are as similar as possible to $\hat{x}_i$ using the Levenshtein distance comparison principle. For example, for the token "книга", the set of variants can be V = {{"книга"}, {"книги"}, {"книзі"}}. To identify grammatical errors in accordance with the rules of Ukrainian grammar, the system uses morphological analysis. This limits the list of candidates to words that match the required format. Language models, in turn, determine context-sensitive candidates using pre-trained models that consider the context of neighboring words when generating corrections.
4. Candidate evaluation. For each candidate $y_i \in D$, the system calculates the probability of matching the context. The evaluation is carried out using language models (BERT, GPT) or grammar rules. The models calculate which replacement y_i is the best in the context of the entire sentence. In the sentence "Я люблю читати книгі", the probability P({"книги" | "Я люблю читати"}) will be higher than for "книзі". The grammar rules calculate additional constraints on the choice of words.
5. Choosing the best candidate. For an erroneous token x_i, the candidate $\mathring{v}_j$ with the maximum probability score is chosen:

$$\mathring{v}_j = \arg\max_{v_j \in D} P\left(v_j | context\right), \tag{17}$$

   For "книгі", the value is $\mathring{v}_j$ = {" книги "}, since this is the most likely option in the given context.
6. Correction of punctuation errors. If the error type y_i indicates a punctuation violation, then instead of the token x_i, the correct punctuation mark defined by the syntax rules is inserted. For the text "Не знаю як пояснити", the system will correct it to "Не знаю, як пояснити".

7. Formation of the final text. After processing each token, the source text is formed. If the token was not changed, it is added to the final list $\hat{X}$ in its original form. If the token was changed, the corrected version $\mathring{v}_j$ is added to the list $\hat{X}$.
8. Checking consistency. For the entire sentence, a final check of consistency between tokens is performed: checking the grammatical correctness of the agreement between the subject and the predicate, nouns and adjectives, verbs and their arguments; checking the semantic content correspondence of the entire construction.
9. Returning the corrected text. The system returns the corrected text $\hat{X}$, and can also provide a list of changes made with an indication of the type of each error.

To visualize the algorithm and processes of the error correction module in Ukrainian-language texts, DFD diagrams were constructed.

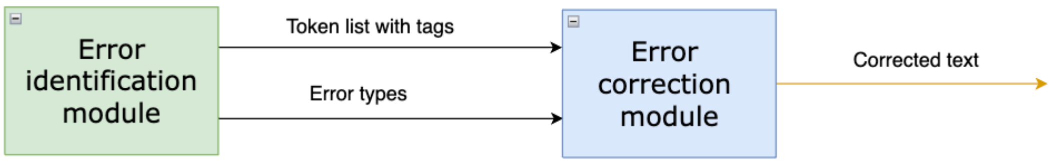


Figure 6: DFD diagram level 0 of the error correction module

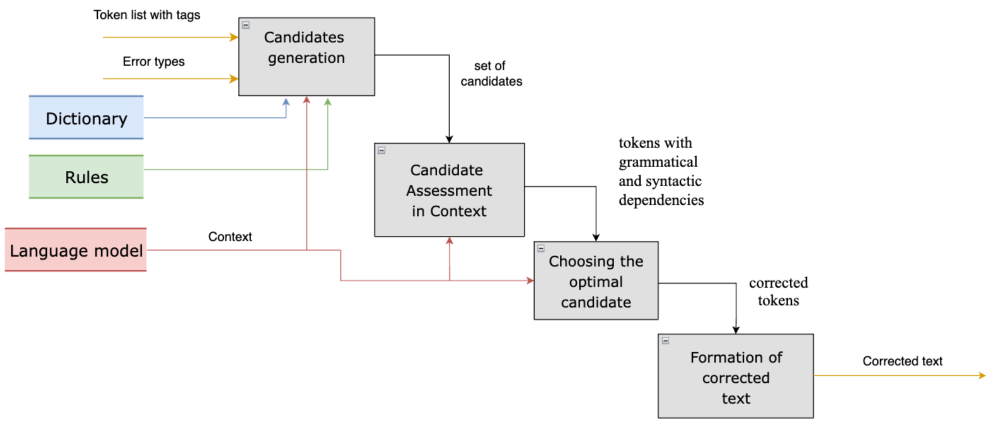

Token list with tags
Error types
Candidates generation
set of candidates
Dictionary
Rules
Candidate Assessment in Context
tokens with grammatical and syntactic dependencies
Language model
Context
Choosing the optimal candidate
corrected tokens
Formation of corrected text
Corrected text

Figure 7: Level 1 DFD diagram for the error correction module.

Example of the algorithm with the input text "Я люблю читати книгі й розкажу друзі про цю книгу.":

Labels from the identification module: {{"Я": no errors}, {"люблю": no errors}, {"читати": no errors}, {"книгі": spelling}, {"й": no errors}, {"розкажу": no errors}, {"друзі": grammatical}, {"про": no errors}, {"цю": no errors}, {"книгу": no errors} }. Correction: "книгі" $\longrightarrow$ "книги", "друзі" $\longrightarrow$ "друзям".

Corrected text: " Я люблю читати книги й розкажу друзям про цю книгу."

7. Mathematical model of machine learning

The text in its original form is not suitable for processing by machine learning algorithms that work with numerical data. Therefore, the main task of this stage is to convert the text into a format that allows it to be used to build models that can learn effectively. Before starting the formation of vector representations of the text, the tokenization process is performed, which must consider the peculiarities of Ukrainian grammar, such as the processing of complex words, particles, and quotes.

After tokenization, each token must be converted into a numerical format suitable for machine processing. Depending on the task and the methods used to process the text, each token can be represented in different formats: as a number in a dictionary corresponding to a specific token (integer encoding), as a multidimensional representation (One-Hot Encoding), or as a word embeddings. The first two methods do not scale for large corpora and do not consider the relationships between tokens, which makes them not the best option for selection.

Word embeddings are multidimensional vectors that consider semantic and syntactic relationships between words and are built based on statistical models. The most common methods are Word2Vec, FastText, and BERT. Word2Vec is a statistical method for constructing vectors through training on text corpora. Vectors of similar words are closer to each other in a multidimensional space. For example: {"собака"} ≈ [0.2, 0.8, -0.3], {"вовк"} ≈ [0.1, 0.7, -0.4]. FastText is an extension of Word2Vec that considers sub-word elements (morphemes) within a token. This is important for the Ukrainian language due to its high morphological variability. BERT, in turn, generates contextual vectors that depend on the position of the word in the sentence. For example: "сльози" in the sentence "сльози на очах" will have a different vector than "сльози природи".

Each token x_i from the text X can be represented as a vector W. The training sample represents the data on which the model will learn to recognize errors and suggest corrections. When working with Ukrainian-language texts, it is important to consider not only the general principles of data collection, but also the specifics of the Ukrainian language, which includes rich morphology, complex syntactic and grammatical structure, as well as the presence of regionalisms, borrowings and colloquial speech.

The main tasks of this stage are: creating a representative corpus of texts covering different aspects of the language: formal, colloquial and artistic styles; introducing artificial errors that simulate real types of errors in the text, or using corpora with errors marked manually; balancing the data to avoid the dominance of one type of error and ensure uniform training of the model; building a data structure that combines the original text, modified text with errors, error labels and the correct version of the text.

To train the model, a corpus of texts containing marked language errors is required. Sources of such corpora: real texts, including student essays, scientific papers with errors, social networks, blogs, personal messages, online forums and annotated corpora, which are existing corpora for the Ukrainian language (UA GEC, GRAC [16] or similar), where the texts have already been annotated by experts for correction tasks. Error-free texts are also important for building the training sample. They are used to introduce artificial errors to simulate real scenarios and train the model to correctly "read" the context and identify non-problematic areas of the text. Here, the sources can be books, formal documents, news; Wikipedia and other open text databases; works of art in the Ukrainian language.

The training sample should represent a variety of error types. The main categories include spelling (randomly added letters, replacing one letter with another, omission or excess of letters), grammatical (incorrect word agreement, conjugation or verb conjugation errors, incorrect word order), punctuation, lexical and contextual errors. The question of increasing the size of the training sample often arises. To do this, you can use automatic error generation using rules or static algorithms, or ready-made language models (GPT, mT5) to generate texts with typical errors. For automatic error generation, you can use an algorithm for random error insertion, replacing letters based on the frequency of real errors (for example, "с" ⟶ "з"), deleting or adding characters to words, omitting commas or adding unnecessary punctuation marks. It is also often used to replace the correct case with a random one ("до школи" ⟶ "до школою") or randomly add words out of context ("Я граю в футбол." ⟶ "Я гамак в футбол.").

To ensure high-quality training of the model, it is necessary to create a balanced sample, where each type of error should have enough examples and at the same time the ratio of incorrect texts to correct ones should not be excessively large (40% of incorrect texts to 60% of correct ones). To create a high-quality sample, the texts should be annotated manually. To do this, each error is marked with the appropriate type, the exact positions of the incorrect tokens are indicated, and the correct version of the text is added next to the errors. Result: each text will become an object of the format: {"Я іду чтати книгу."} ⟶ {[1, {"іду","grammatical error","йду"}], [2, {"чтати","spelling error”,"читати" }]}. As a result, the corpus of texts is divided into three parts: a training sample for building the model (~70% of the data), a validation sample for checking the quality during training (~15% of the data), and a test sample for assessing accuracy on new, unknown texts for the model (~15% of the data).

The process of training a machine learning model for the problem of error identification and correction includes setting up the model architecture, defining the objective function, optimizing the parameters, as well as checking and validating the obtained results. Model should maximize the probabilities of:

$$P(Y|X)=\prod_{i=1}^{N} P(y_i|x_1,x_2,\ldots,x_N), \tag{18}$$

$$P(\hat{X}|X,Y)=\prod_{i=1}^{N} P(\hat{x}_i|x_1,x_2,\ldots,x_N,y_1,y_2,\ldots,y_N), \tag{19}$$

where $P(\hat{x}_i|X,Y)$is the probability of correcting the token x_i, $P(y_i|X)$is the probability that the token $x_i$belongs to the class y_i. Training involves the use of modern text processing methods based on neural language models. For the problem of error identification and correction, two architectures are most effective: recurrent neural networks (RNN, LSTM, GRU) and Transformer. Recurrent neural networks are used for the problem of sequence analysis. Their main advantage is that they preserve the context of previous tokens. For example, LSTM (Long Short-Term Memory) allows you to consider long-term dependencies between tokens:

$$h_t=LSTM(x_t,h_{t-1}), \tag{20}$$

where h_t is the hidden state at step t, which contains information about the current token and the previous ones.

Transformer models such as BERT, XLM-RoBERTa or mT5 are the most effective for identifying and correcting errors in texts. Thanks to the attention mechanism, dependencies between all tokens in the text are considered, regardless of their positional distance. BERT is used for error identification, and mT5 or GPT can be used for error correction.

The objective function determines how well the model recognizes errors and suggests correct corrections. For the error identification problem, the loss function is based on cross-entropy:

$$Loss_{id}=-\frac{1}{N}\sum_{i=1}^{N}\sum_{k=0}^{K} y_{(i,k)}\log\left(\hat{y_{(i,k)}}\right), \tag{21}$$

where: $y_{(i,k)}$ is the label of token x_i belonging to class c_k; $\hat{y_{(i,k)}}$ – is the probability predicted by the model for this class. For the error correction problem, the loss function estimates the difference between the corrected text and the correct text

$$Loss_{cor}=-\frac{1}{N}\sum_{i=1}^{T}\log\left(P(\hat{x}_i,X,Y)\right), \tag{22}$$

where $P(\hat{x}_i,X,Y)$ is calculated using the transformer model. The overall objective function of the model looks like this:

$$Loss=\lambda_1\cdot Loss_{id}+\lambda_2\cdot Loss_{cor} \tag{23}$$

where $\lambda_1$and λ_2 are the weighting factors for balancing the tasks.

The learning algorithm can be described as follows:

1. Model initialization. At this stage, the model architecture is selected and hyperparameters are configured (number of layers, number of neurons, embedding size, etc.).
2. Data preparation. The texts are converted into vector representations, and the training sample is divided into three subsets: training, validation, and test (70% / 15% / 15%).
3. The learning process. Here, tokens from the sample are fed to the model input, the predicted error classes $\hat{Y}$ and corrected texts $\hat{X}$ are calculated. Next, the process of calculating the loss value Loss takes place. At the finish of this stage, the model parameters are optimized using gradient descent algorithms (Adam or AdamW).
4. Validation. After each epoch, the learning process is checked on validation data. The model is evaluated using the following metrics: Accuracy, Precision, Recall, F1-score, and BLEU to assess the quality of corrected texts.
5. Testing. Model evaluation on the remaining test data to obtain final metrics.

After training, the model becomes capable of accurately classifying text tokens by error type, offering high-quality corrections that are consistent with the context, and generating corrected text with minimal stylistic and grammatical violations.

8. Model fine-tuning

In this study, the ‘facebook/mbart-large-50’ model was chosen for the task of automatic error correction in Ukrainian-language texts. The choice of this transformer architecture is due to its multilingual nature, which allows it to work effectively with less resource-intensive languages, particularly Ukrainian. Due to pre-training on many parallel texts, MBART has a high potential for generating grammatically and stylistically correct sentences in Seq2Seq tasks. In addition, the model performs well in other text transformation tasks, in particular paraphrasing and machine translation, which are close in nature to error correction.

The UA-GEC corpus was used for training, which contains examples of sentences from typical errors in the Ukrainian language and their corrections. Data pre-processing was carried out using the Stanza library, for dividing into source (input, erroneous sentences) and target (corrected sentences), which corresponded to the Seq2Seq training format.

The training was performed in the Kaggle environment with an available GPU and limited RAM. Considering the resource constraints, a compromise set of hyperparameters was selected: batch_size = 4, max_token_length = 64, learning_rate = 5e-5, n_epochs = 6. The training was performed in two stages: first 3 epochs with basic settings, and then additional retraining for another 3 epochs with a lower learning rate. Initially, the training rate was 5e-5, which contributed to the rapid assimilation of the main patterns, while reducing the rate to 3e-5 in the later stages helped to avoid overtraining and allowed for more precise adjustment of the weights.

Table 1
The hyperparameters for model training over iterations

Hyperparameter	Iteration №1	Iteration №2
Epoch count	3	3
Batch size	4	4
Maximum token length	64	64
Learning rate	5e-5	3e-5

As part of the quantitative analysis of the quality of the model for correcting errors in Ukrainian texts, several metrics were used that allow an objective assessment of the learning dynamics. In particular, the values of the text-level metrics sacreBLEU, BLEU, and METEOR were recorded, as well as the losses on the training and validation samples during three epochs of learning.

The initial quality of the model, recorded in the first epoch, turned out to be expectedly low. The value of the BLEU metric was zero, sacreBLEU was only 0.004, and METEOR was 0.011. This indicates the almost complete inability of the model at the start to generate any corrections that at least partially coincided with the reference ones. Such a result is typical for transformer models until they begin to form meaningful correspondences between the input and target sequences.

However, a significant breakthrough is observed already in the second epoch. The BLEU value increases sharply to 0.477, sacreBLEU to 47.76, and METEOR to 0.584. This indicates that the model has begun to effectively learn the dependencies between incorrect and correct sentences, reproducing a significant part of the target responses either with an exact match or with a high degree of semantic similarity. It is also important to note the drop in the value of the loss function: the training loss decreased from 4.16 to 2.71, and the validation loss from 3.70 to 0.99, which is a good indicator that the learning is going in the right direction, and the model not only remembers but also generalizes the patterns.

In the third epoch, the positive dynamics remains: BLEU reaches 0.659, sacreBLEU – 65.92, METEOR – 0.687. At the same time, the losses are reduced even more significantly: the training loss drops to 0.43, the validation loss to 0.51. This indicates the stability of the learning process and the absence of signs of overtraining, which is especially important for the text correction task, where the generative ability of the model must be flexible, not highly specialized.

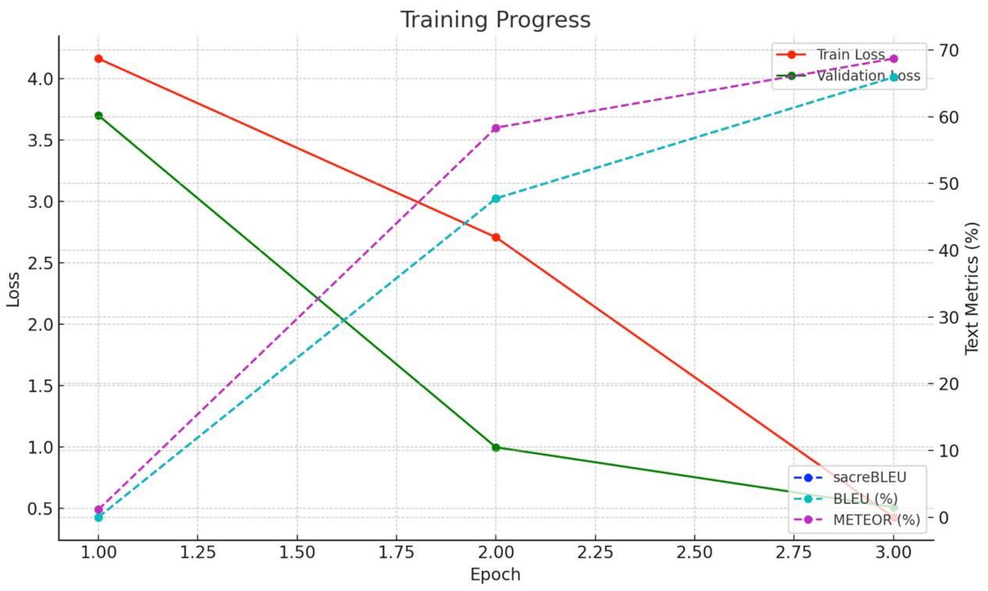


Figure 8: Loss graph considering the BLUE and METEOR metrics.

Starting from the fourth epoch, training continued at a slower learning rate – learning_rate was reduced from 5e-5 to 3e-5. This allowed the model to more accurately “finish” the already formed correspondences, but the quality gain was insignificant: BLEU increased to 0.662, METEOR – to 0.692, and sacreBLEU – to 66.16. Such stability of metrics indicates that the model has reached a plateau – it has already acquired the basic regularities and now only slightly refines the predictions. At the same time, the validation loss began to slowly increase (from 0.51 to 0.57), which may indicate the beginning of overtraining. Thus, reducing learning_rate made it possible to avoid sharp fluctuations, but did not provide a significant improvement – that is, the best results were at 3-4 epochs.

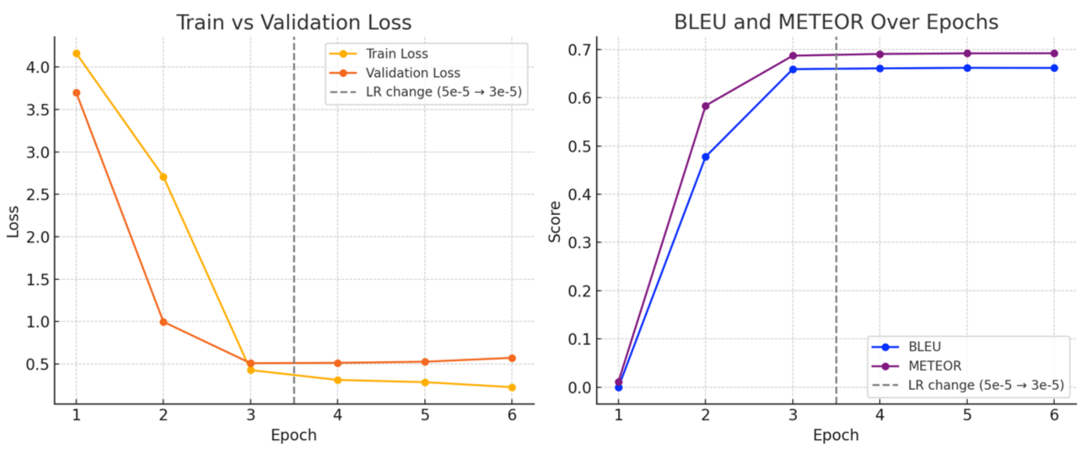
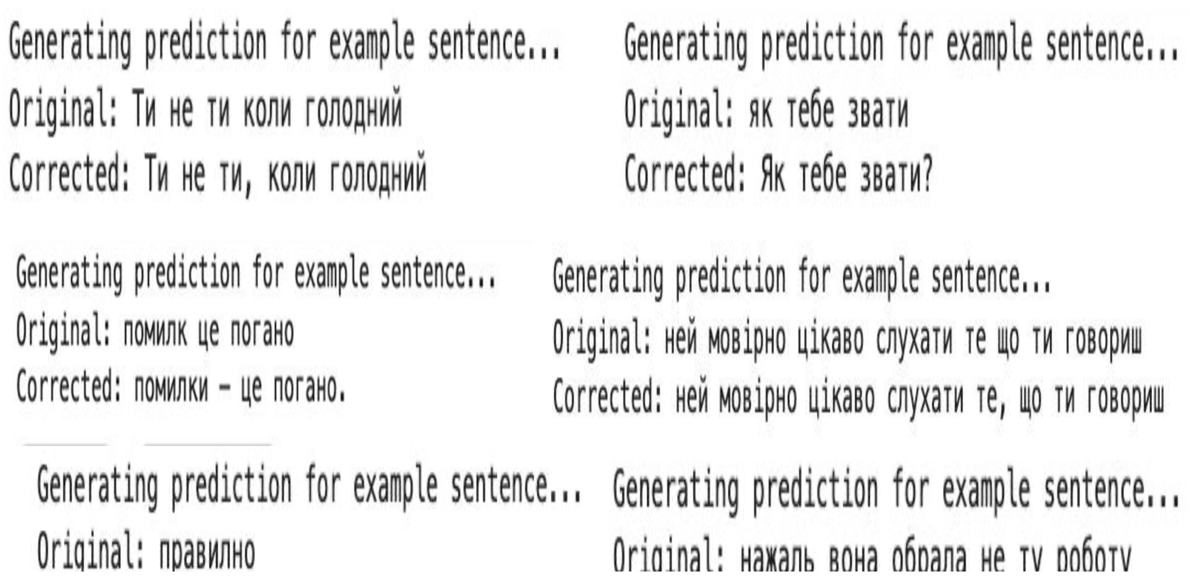


Figure 9: Loss graph and graph of BLUE and METEOR metrics after changing the learning rate parameter.

Summarizing these graphs, we can conclude that the chosen approach to training the model is effective. The high indicators of sacreBLEU and METEOR demonstrate not only lexical accuracy, but also stylistic adequacy of the generated variants. The rapid growth of metrics between the first and second epochs indicates that even in a short training time the model can master the basics of language correction. In general, this gives grounds for a confident forecast regarding the potential improvement of quality with further training, expansion of the training corpus, and enrichment of the types of language errors in the input examples. The quality of the model was assessed by manual testing on a set of sentences with typically common errors that often occur in Ukrainian-language texts. In total, dozens of sentences were tested, covering various types of language errors, including punctuation, spelling, lexical, and morphological. Analysis of the results allows us to draw conclusions about the current level of formed linguistic generalizations in the model, identify its strengths, and identify areas where there is a lack of linguistic competence.


```
Generating prediction for example sentence...
Original: Ти не ти коли голодний
Corrected: Ти не ти, коли голодний
```

```
Generating prediction for example sentence...
Original: як тебе звати
Corrected: Як тебе звати?
```

```
Generating prediction for example sentence...
Original: помилк це погано
Corrected: помилки - це погано.
```

```
Generating prediction for example sentence...
Original: ней мовірно цікаво слухати те що ти говориш
Corrected: ней мовірно цікаво слухати те, що ти говориш
```

```
Generating prediction for example sentence...
Original: правилно
```

```
Generating prediction for example sentence...
Original: нажаль вона обрала не ту роботу
```

Figure 10: Results of the model after training.

The model performed best in correcting punctuation errors. In most cases, it confidently recognized and corrected the absence of commas in complex constructions. For example, the sentence “Я бачив як вона грає на піанино” was successfully converted to “Я бачив, як вона грає на піанино” and in the sentence “Коли я приїхав у Львів мені сподобалось атмосфера” a comma was added after the subordinate clause, which complies with the norms of Ukrainian syntax. Similarly, in a sentence with direct speech, the model used the correct punctuation – “не хай так і буде сказав дмитро” was corrected to “Не хай так і буде, – сказав дмитро”, which demonstrates the model’s understanding of the specifics of direct speech and its intonation highlighting in writing, although the model did not provide quotation marks.

However, the system only partially coped with spelling errors. Simple cases, such as misspelling a word with a space ("Близ ько") or using a lowercase service word at the beginning of a sentence, were successfully corrected. But in more complex situations, where correction requires a deeper analysis of the dictionary form, the system failed. For example, the word "ней мовірно" remained unchanged, although it should have been corrected to "неймовірно". This indicates that the current version of the model does not always have a sufficient level of spelling sensitivity, especially in cases where errors do not have clear contextual clues and require checking for compliance with the dictionary norm.

A similar situation is observed with morphological errors. In the sentence " Коли я приїхав у Львів, мені сподобалось атмосфера" the model did not detect a conflict between the neuter verb and the feminine noun, leaving the erroneous form unchanged. This indicates an insufficient depth of coordination between grammatical categories in the processing of the input text.

In rare cases, the model demonstrates the ability to perform syntactic transformations, changing the structure of a sentence from incorrect to grammatically correct. For example, the sentence “я тебе звати” was transformed into “Я тебе зватиму”, indicating the formation of a basic understanding of predicative constructions and verb agreement. However, most examples of this type remain beyond the scope of successful correction, and the transformational ability of the model currently appears limited.

9. Conclusions

The result of the work is a developed mathematical model of a decision support system for identifying and correcting errors in Ukrainian-language texts, focused on the implementation of machine learning approaches. As a result of the research, a mathematical basis was formulated for solving the tasks set, including the formalization of the data flow, the placement of system components, and the presentation of texts in a form suitable for machine processing.

The main mathematical structure of the system was highlighted, which consists of two key modules: for identifying and correcting errors. Both modules interact within the system, ensuring the correct processing of the text sequence. A separate mathematical model was built for the problem of identifying errors, which is based on a probabilistic approach. The main emphasis in modeling is placed on preserving the context and considering dependencies between tokens to increase the accuracy of identification. A mathematical model was built that involves calculating the conditional probability of choosing the best candidate for correcting an erroneous token within a given text context. The approach to data preparation through text vectorization, formation of a training sample and organization of the model training process based on large text corpora was considered. The importance of building a representative corpus of training data, which includes texts with real errors, as well as artificially simulated examples of errors considering their typological distribution, was emphasized.

The model was trained on the UA_GEC corpus, which demonstrated encouraging results in correcting punctuation and basic spelling errors, especially within simple and clearly structured sentences. At the same time, it is not effective enough in detecting lexical and morphological errors, and still poorly copes with deeper syntactic rearrangements. The results outline clear directions for further improvement of the system, namely, expanding the training corpus with examples with morphological and lexical errors, as well as introducing additional processing mechanisms that would compensate for the limited orthographic competence of the model. In the future, improving these aspects will allow creating a more reliable and comprehensive tool for automatic verification of Ukrainian-language texts.

As a result, the developed mathematical model is a universal approach to processing Ukrainian texts, which allows solving the problems of identifying and correcting errors within a single system. The determined formal aspects of the interaction between the components of the model create the basis for its effective training and further implementation in practical tasks.

Acknowledgements

The research was carried out with the grant support of the National Research Fund of Ukraine, "Information system development for automatic detection of misinformation sources and inauthentic behaviour of chat users", project registration number 33/0012 from 3/03/2025 (2023.04/0012). Also, we would like to thank the reviewers for their precise and concise recommendations that improved the presentation of the results obtained.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-4 in order to: Grammar and spelling check. After using these tool, the authors reviewed and edited the content as needed and take full responsibility for the publication’s content.

References

[1] C. Bryant, Z. Yuan, M. R. Qorib, H. Cao, H. T. Ng, T. Briscoe, Grammatical Error Correction: A Survey of the State of the Art, Comput. Linguist. 49 (3) (2023) 643–701. doi:10.48550/arXiv.2211.05166.

[2] O. B. Smith, J. O. Ilori, P. Onesirosan, The proximate composition and nutritive value of the winged bean Psophocarpus tetragonolobus (L.) DC for broilers, Anim. Feed Sci. Technol. 11 (1984) 231–237.

[3] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, C. Raffel, mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer, in: Proc. Conf. NAACL: HLT, Online, 2021, pp. 483–498. doi:10.18653/v1/2021.naacl-main.41.

[4] Grammarly Inc., About Us, Grammarly Inc. (n.d.). URL: https://www.grammarly.com/about.

[5] Grammarly, UA-GEC. URL: https://github.com/grammarly/ua-gec.

[6] O. Syvokon, O. Nahorna, UA-GEC: Grammatical Error Correction and Fluency Corpus for the Ukrainian Language, 2021. doi:10.48550/arXiv.2103.16997.

[7] M. Brovinska, “I waited eight years for Grammarly to support Ukrainian.” A Ukrainian created a tool for spell checking and translating in two clicks. It supports 76 languages, URL: https://dev.ua/news/ai-servisy-1706885687.

[8] LanguageTool, We believe that anyone can write beautifully and professionally. URL: https://languagetool.org/about.

[9] LanguageTool Community, Error Rules for LanguageTool, URL: https://community.languagetool.org/rule/list?offset=0&max=10&lang=uk&filter=&categoryFilter=&_action_list=%D0%A4%D1%96%D0%BB%D1%8C%D1%82%D1%80.

[10] Github, LanguageTool API NLP UK, URL: https://github.com/brown-uk/nlp_uk.

[11] OnlineCorrector, Writing correctly is easy, URL: https://onlinecorrector.com.ua

[12] K. Omelianchuk, V. Atrasevych, A. N. Chernodub, O. Skurzhanskyi, GECToR – Grammatical Error Correction: Tag, Not Rewrite, ArXiv, 2020. doi:10.48550/arXiv.2005.12592.

[13] HuggingFace, Grammatical Error Correction Models, URL: https://huggingface.co/models?other=grammatical+error+correction&sort=trending&search=gec.

[14] A. Katinskaia, R. Yangarber, GPT-3.5 for Grammatical Error Correction, 2024. doi:10.48550/arXiv.2405.08469.

[15] A. Luhtaru, E. Korotkova, M. Fishel, No Error Left Behind: Multilingual Grammatical Error Correction with Pre-trained Translation Models, in: Proc. EACL 2024, Vol. 1: Long Papers, St. Julian's, Malta, 2024, pp. 1209–1222.

[16] M. Shvedova, et al., General Regionally Annotated Corpus of Ukrainian Language (GRAC), Network for Ukrainian Studies Jena (2017–2022). doi:10.48550/arXiv.1911.02116.

[17] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a Method for Automatic Evaluation of Machine Translation, in: Proc. 40th Annual Meeting of ACL, Philadelphia, USA, 2002, pp. 311–318. doi:10.3115/1073083.1073135.

[18] V. Lytvyn, P. Pukach, V. Vysotska, M. Vovk, N. Kholodna, Identification and correction of grammatical errors in Ukrainian texts based on machine learning technology, Mathematics 11 (4) (2023) 904. doi:10.3390/math11040904.

[19] H. Lipyanina, A. Sachenko, T. Lendyuk, S. Nadvynychny, S. Grodskyi, Decision tree based targeting model of customer interaction with business page, in: CMIS-2020 Computer Modeling and Intelligent Systems, CEUR Workshop Proc., vol. 2608, CEUR-WS.org, 2020. urn:nbn:de:0074-2608-1.

[20] A. Chiche, Hybrid decision support system framework for crop yield prediction and recommendation, Int. J. Comput. 18 (2) (2019) 181–190. doi:10.47839/ijc.18.2.1416.

[21] O. Mediakov, V. Vysotska, D. Uhryn, Y. Ushenko, C. Hu, Information technology for generating lyrics for song extensions based on transformers, Int. J. Mod. Educ. Comput. Sci. 16 (1) (2024) 23–36.

[22] H. Lipyanina, V. Maksymovych, A. Sachenko, T. Lendyuk, A. Fomenko, I. Kit, Assessing the investment risk of virtual IT company based on machine learning, in: S. Babichev, D. Peleshko, O. Vynokurova (Eds.), Data Stream Mining & Processing. DSMP 2020, Communications in Computer and Information Science, vol. 1158, Springer, Cham, 2020, pp. 122–134. doi:10.1007/978-3-030-61656-4_11.

[23] S. Bhatia, M. Sharma, K.K. Bhatia, P. Das, Opinion target extraction with sentiment analysis, Int. J. Comput. 17 (3) (2018) 136–142. doi:10.47839/ijc.17.3.1033.

[24] A. Demchuk, B. Rusyn, L. Pohreluk, et al., Commercial content distribution system based neural network and machine learning, in: CEUR Workshop Proc., 2019, pp. 40–57.