

## ДОДАТОК А

### Програмний код вебдодатка

#### Лістинг А.1 – Програмний код файлу main.py

```
from fastapi import FastAPI, Request
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from fastapi.responses import JSONResponse
import language_tool_python
import os
import nltk
nltk.download('punkt')
os.environ["TRANSFORMERS_NO_TF"] = "1"
from transformers import pipeline
import requests
import numpy as np
import pandas as pd
from sklearn.pipeline import Pipeline
from lime.lime_text import LimeTextExplainer
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import
TfidfVectorizer
import json

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

## Продовження лістингу A.1

```

OPENROUTER_API_KEY = "" # ключ приховано в цілях безпеки

corrector = pipeline("text2text-generation",
model="vennify/t5-base-grammar-correction")

class TextRequest(BaseModel):
    text: str

class ExplainRequest(BaseModel):
    language: str

def correction_languagetool(text):
    tool = language_tool_python.LanguageTool('en-US')
    matches = tool.check(text)
    corrected_text =
language_tool_python.utils.correct(text, matches)
    return corrected_text

def correction_gpt(text):
    prompt = f"""
        You are an English writing assistant. The user will
        provide you with a text that contains grammatical,
        spelling, punctuation, and lexical mistakes.

        Your task:
        1. Correct only lexical and semantic mistakes in the
        text, if there are any. Make sure all the sentences make
        sence separately and as the whole text. Pick up the style
        of the text - formal, informal or something in between.
        Make corrections according to that style - for example, if
        the text is clearly informal, do not turn the informal
        frases into formal ones.
        2. Output only the corrected version at the end. Do
        not include the word \"corrected\" at the beginning.
    """

```

## Продовження лістингу А.1

```
Text: {text}
"""

headers = {
    "Authorization": f"Bearer {OPENROUTER_API_KEY}",
    "Content-Type": "application/json",
    "HTTP-Referer": "https://yourdomain.com",
    "X-Title": "text-correction-app"
}

data = {
    "model": "openai/gpt-4o-2024-11-20",
    "messages": [
        {"role": "user", "content": prompt}
    ]
}

response = requests.post(
    "https://openrouter.ai/api/v1/chat/completions",
    headers=headers,
    json=data
)

try:
    res_json = response.json()
    if 'choices' not in res_json:
        return "⚠️ Виникла помилка під час обробки
відповіді."
    corrected_text =
res_json['choices'][0]['message']['content']
    return corrected_text.strip()
except Exception as e:
    print("⚠️ GPT ERROR:", str(e))
```

## Продовження лістингу А.1

```

        return "⚠️ Виникла помилка під час обробки
відповіді."

def correct_text_by_sentences(text, max_length=128):
    sentences = nltk.sent_tokenize(text)
    corrected_sentences = []

    for sent in sentences:
        correct_grammar = corrector(
            sent,
            max_length=max_length,
            do_sample=False,
            num_beams=4,
            early_stopping=True
        )
        correct_spelling =
correction_languagetool(correct_grammar[0]['generated_text
'])
        correct_semantic =
correction_gpt(correct_spelling)
        corrected_sentences.append(correct_semantic)

    return " ".join(corrected_sentences)

@app.post("/correct")
def correct_and_explain(req: TextRequest):
    start_text = req.text
    correct_text = correct_text_by_sentences(start_text)

    texts = [start_text, correct_text]
    labels = [1, 0]

    vectorizer = TfidfVectorizer()
    classifier = LogisticRegression()

```

## Продовження лістингу А.1

```

X = vectorizer.fit_transform(texts)
classifier.fit(X, labels)

pipe = make_pipeline(vectorizer, classifier)

explainer = LimeTextExplainer(class_names=["correct",
"error"])
explanation = explainer.explain_instance(
    start_text,
    pipe.predict_proba,
    num_features=6
)

features = explanation.as_list()
lime_explanation_json = {
    "text": start_text,
    "top_tokens": [
        {"token": word, "importance": score}
        for word, score in features
    ],
    "predicted_label":
int(pipe.predict([start_text])[0]),
    "class_names": explanation.class_names,
}

# Зберігаємо LIME JSON
with open("lime_explanation.json", "w", encoding="utf-
8") as f:
    json.dump(lime_explanation_json, f, indent=4)

# Зберігаємо виправлений текст
with open("corrected_text.txt", "w", encoding="utf-8")
as f:
    f.write(correct_text)

```

## Продовження лістингу A.1

```

    # Зберігаємо HTML з LIME
    with open("text_explanation.html", "w", encoding="utf-
8") as f:
        f.write(explanation.as_html())

    return JsonResponse({
        "corrected_text": correct_text,
        "lime_result": lime_explanation_json
    })

class ExplainRequest(BaseModel):
    language: str

def generate_explanation_prompt(json_data, corrected_text,
language="en"):

    text = json_data.get("text", "")
    predicted_label = json_data.get("predicted_label", "")
    class_names = json_data.get("class_names", [])
    features = json_data.get("top_tokens", [])

    label_name = class_names[predicted_label] if
class_names and predicted_label < len(class_names) else
str(predicted_label)
    important_features = [f for f in features if
abs(f.get("importance", 0)) > 0]

    explanation_lines = [
        "You are an expert in English linguistics and
writing.",
        "A machine learning model analyzed the following
sentence and identified potentially problematic or
incorrect parts.",
        "",

```

## Продовження лістингу A.1

```

    f"Text input from the user is: \"{text}\"",
    f"The corrected version is:\n{corrected_text}",
    f"🤖 Model's prediction: {label_name.lower()}",
    "",
    "The XAI method used to explain the model's
decision is LIME. It considers the following tokens
important for the correction model's decision:"
]

if not important_features:
    explanation_lines.append("None - the model didn't
find any strongly influencing words.")
else:
    for i, feature in enumerate(important_features,
1):
        token = feature.get("token", "")
        weight = feature.get("importance", 0)
        explanation_lines.append(f"{i}. '{token}'
(importance score: {weight:.4f})")

explanation_lines.append("")
explanation_lines.append(
    f"Please explain in English why these tokens might
indicate that the sentence is incorrect or correct."
    f"Compare the initial text and the corrected one,
use the differences to explain the corrections, as well as
your own knowledge of the language and common sense. Do
not make up other corrections, only rely on differences
between two texts and the LIME's explanation."
    f"Use the simple language, explain as an English
language teacher to a student of a medium-level knowledge.
Do not include technical details in the explanation."

```

## Продовження лістингу A.1

```
f"Explain every token LIME found important from
the list above, make it an unnumbered list. Include a
small introduction and conclusion to sound more natural."
```

```
f"Focus on grammar, semantics, or other writing
issues the model may have found."
```

```
f"Explain using rules of english language but keep
the explanations brief and simple."
```

```
f'Format your explanation using HTML instead of
Markdown. Use <strong> for bold and <h4> or <p> for
structure. Do not use multiple <br> tags for spacing -
instead, use <p> or <div> with minimal bottom margin. Do
not use <br> between <h4> and <p>. Wrap the entire output
in <div style="font-family: Arial, sans-serif; line-
height: 1.4; margin: 0; padding: 0;">.\n'
```

```
)
```

```
return "\n".join(explanation_lines)
```

```
@app.post("/explain")
```

```
def explain(req: ExplainRequest):
```

```
    lang = req.language.lower()
```

```
    try:
```

```
        with open("lime_explanation.json", "r",
encoding="utf-8") as f:
```

```
            json_data = json.load(f)
```

```
    except FileNotFoundError:
```

```
        return JsonResponse(content={"explanation": "⚠️
No LIME explanation found."})
```

```
    try:
```

```
        with open("corrected_text.txt", "r",
encoding="utf-8") as f:
```

```
            corrected_text = f.read()
```

## Продовження лістингу А.1

```

except FileNotFoundError:
    corrected_text = ""

    prompt = generate_explanation_prompt(json_data,
corrected_text, language=lang)

    headers = {
        "Authorization": f"Bearer {OPENROUTER_API_KEY}",
        "Content-Type": "application/json",
        "HTTP-Referer": "https://yourdomain.com",
        "X-Title": "lime-text-explainer"
    }

    data = {
        "model": "openai/gpt-4o-2024-11-20",
        "messages": [
            {"role": "user", "content": prompt}
        ]
    }

    try:
        response =
requests.post("https://openrouter.ai/api/v1/chat/completi
ns", headers=headers, json=data, timeout=30)
        gpt_reply =
response.json()["choices"][0]["message"]["content"]
    except Exception as e:
        gpt_reply = f"⚠️ GPT error: {e}"

    try:
        with open("text_explanation.html", "a",
encoding="utf-8") as f:
            f.write("<hr><h3>🧠 Natural Language
Explanation:</h3>")

```

## Продовження лістингу A.1

```

        html_reply = gpt_reply.replace('\n', '<br>')
        f.write(f"<div>{html_reply}</div>")
    except Exception as e:
        print(f"⚠ Could not write GPT explanation: {e}")

    try:
        with open("text_explanation.html", "r",
encoding="utf-8") as f:
            html = f.read()
    except FileNotFoundError:
        html = "⚠ No explanation file found."

    return JsonResponse(content={"explanation": html,
"prompt": prompt})

```

## Лістинг A.2 – Програмний код файлу index.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>TextTune</title>
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <div class="logo">
            
        </div>
        <p>
            Enter the text you want to check for mistakes
and press the arrow.<br>

```

## Продовження лістингу А.2

To get the corrections explained, press  
Explain button.<br>

Please note: both correction and explanation  
take some time, as ML models are working.

</p>

```
<div class="text-box">
  <textarea id="input-text" placeholder="Enter
your text"></textarea>
  <button class="arrow-btn"
onclick="correctText()">→</button>
  <textarea id="output-text"
placeholder="Corrected text will appear here"
readonly></textarea>
</div>

<button class="btn"
onclick="explainCorrections()">Explain</button>

<iframe id="explanation-frame" style="width: 100%;
height: 600px; border: none; display: none;"></iframe>
</div>
<script src="script.js"></script>
</body>
</html>
```

## Лістинг А.3 – Програмний код файлу styles.css

```
body {
  font-family: Arial, sans-serif;
  background-color: #eee;
  text-align: center;
  margin: 0;
  padding: 0;
```

## Продовження лістингу А.3

```
    width: 100%;
    min-height: 100vh;
    box-sizing: border-box;
    overflow-y: auto;
}

.container {
    width: 80%;
    max-width: 1200px;
    min-height: 100vh;
    padding: 20px;
    background: #eee;
    margin: 0 auto;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: flex-start;
}

.logo {
    padding: 10px 20px;
    border-radius: 5px;
    margin-top: 10px;
}

.text-box {
    display: flex;
    justify-content: space-between;
    align-items: center;
    width: 100%;
    margin-top: 20px;
}

textarea {
```

### Продовження лістингу А.3

```
width: 40%;  
height: 300px;  
padding: 10px;  
border: 1px solid #ccc;  
background: #f6f6f6;  
border-radius: 5px;  
resize: none;  
font-size: 16px;  
}
```

```
.arrow-btn {  
padding: 15px 20px;  
font-size: 20px;  
background: lightblue;  
border: none;  
cursor: pointer;  
border-radius: 5px;  
}
```

```
.btn {  
margin-top: 20px;  
padding: 12px 24px;  
background: #90ee90;  
border: none;  
cursor: pointer;  
font-size: 18px;  
border-radius: 5px;  
}
```

### Лістинг А.4 – Програмний код файлу script.js

```
async function correctText() {  
    const inputText = document.getElementById("input-  
text").value.trim();
```

## Продовження лістингу А.4

```

    const outputArea = document.getElementById("output-
text");
    const iframe = document.getElementById("explanation-
frame");

    if (!inputText) {
        outputArea.value = "⚠ Please enter some text.";
        return;
    }

    outputArea.value = "⌚ Correcting...";
    iframe.style.display = "none"; // ховаємо старе
ПОЯСНЕННЯ

    try {
        const response = await
fetch("http://127.0.0.1:8000/correct", {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify({ text: inputText }),
        });

        const data = await response.json();
        outputArea.value = data.corrected_text || "✅
Done, but no correction returned.";
    } catch (error) {
        outputArea.value = "⚠ Error: " + error.message;
    }
}

async function explainCorrections() {

```

## Продовження лістингу А.4

```
    const iframe = document.getElementById("explanation-
frame");
    iframe.style.display = "none";
    iframe.src = "";

    try {
        const response = await
fetch("http://127.0.0.1:8000/explain", {
            method: "POST",
            headers: { "Content-Type": "application/json"
},
            body: JSON.stringify({ language: "en" }) //
фіксована англійська
        });

        const data = await response.json();
        const blob = new Blob([data.explanation], { type:
'text/html' });
        const url = URL.createObjectURL(blob);
        iframe.src = url;
        iframe.style.display = "block";

    } catch (error) {
        alert("⚠ Error generating explanation: " +
error.message);
    }
}
```

## ДОДАТОК Б

### Представлення користувацького інтерфейсу

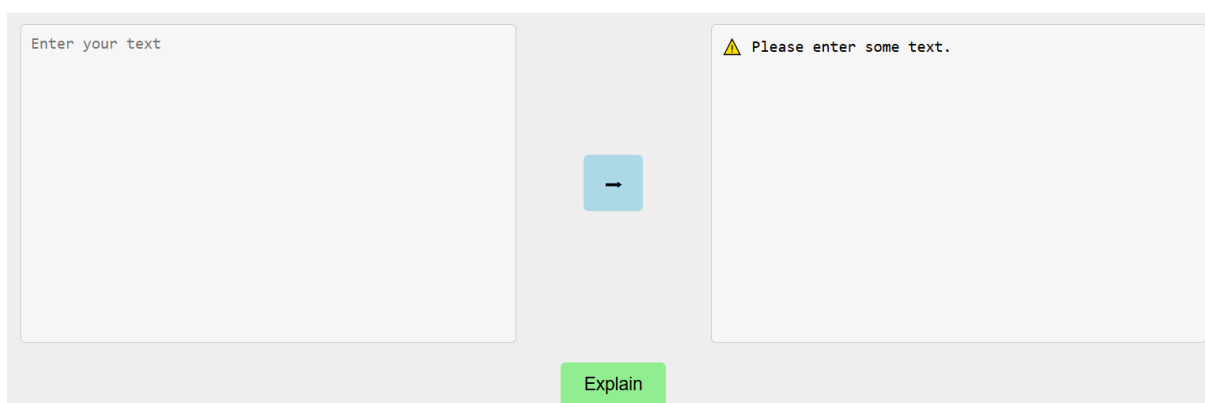


Рисунок Б.1 – Помилка при отриманні порожнього вхідного тексту

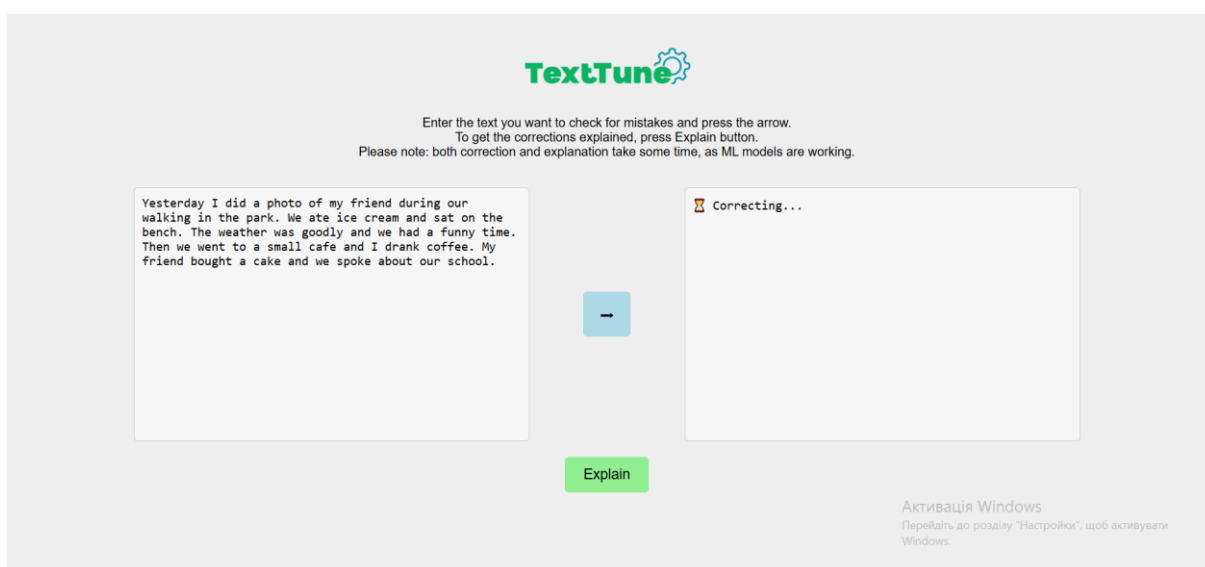


Рисунок Б.2 – Повідомлення про обробку тексту

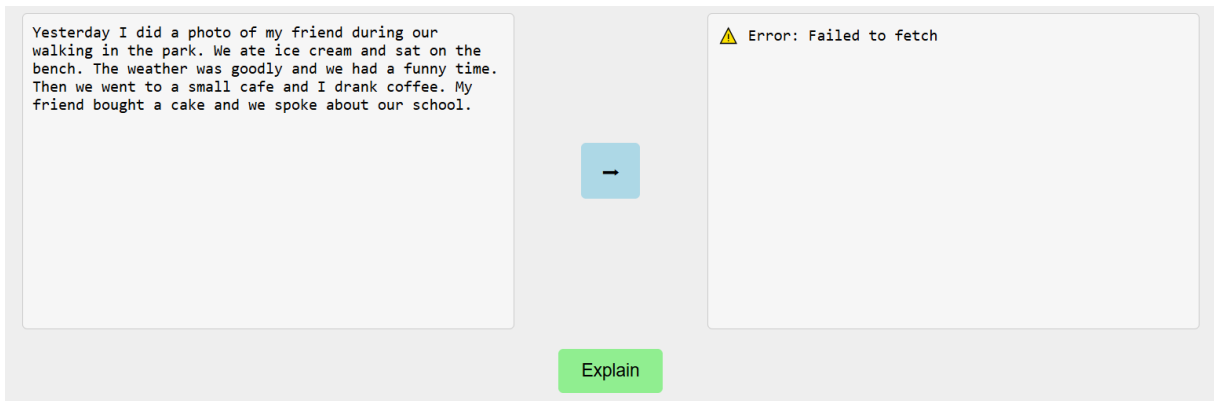


Рисунок Б.3 – Повідомлення про помилку на сервері під час виправлення

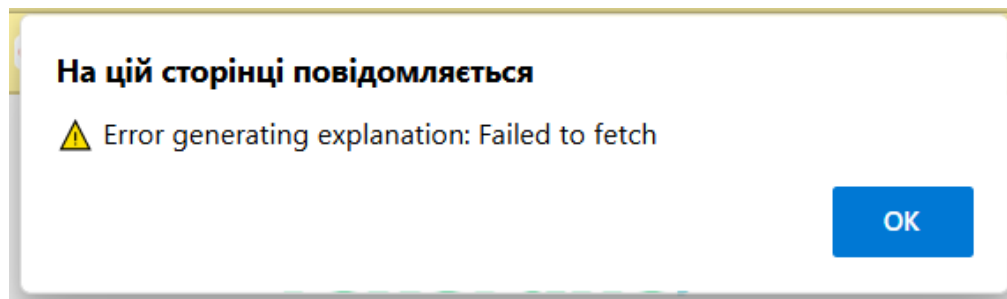


Рисунок Б.4 – Повідомлення про помилку на сервері під час пояснення

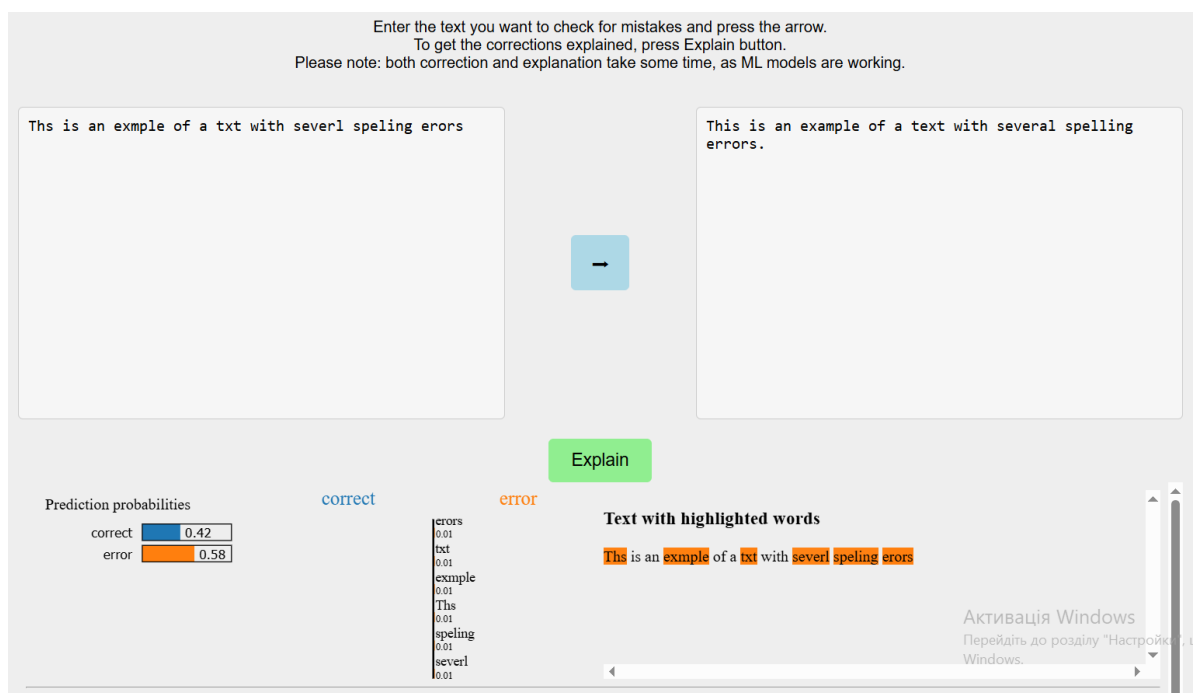


Рисунок Б.5 – Пояснення орфографічних помилок від LIME

**'exmple':** This word is another spelling mistake. In English, the correct spelling is 'example,' with an 'a' between the 'x' and 'm.' Spelling errors like this can make the meaning unclear or seem unprofessional.

**'Ths':** The word 'Ths' in the original text is missing an 'i' to make it the correct word, 'This.' A small error like this can disrupt the clarity and flow of a sentence because 'This' is a very common and important word in English.

**'spelng':** Here, the original word is missing an 'l' to become 'spelling,' which is the correct form. In English, doubling certain consonants for words like this is necessary to meet standard spelling rules. Misspelling this word can confuse readers and affect your writing quality.

**'severi':** The original word 'severi' should be 'several.' This correction adds the missing 'a,' which is an essential part of the word. Leaving it out makes the word incorrect and harder to understand. Proper spelling is important for clear communication.

## Рисунок Б.6 – Приклад пояснення орфографічних помилок природною мовою

Enter the text you want to check for mistakes and press the arrow.  
To get the corrections explained, press Explain button.  
Please note: both correction and explanation take some time, as ML models are working.

He don't like coffee. His friends drinks it every day, but he never tries. Maybe he are allergic to caffeine.



He doesn't like coffee. His friends drink it every day, but he never tries it. Maybe he's sensitive to caffeine.

Prediction probabilities

correct	0.47
error	0.53

correct

allergic  
0.01

arc  
0.01

don  
0.01

drinks  
0.01

it  
0.01

he  
0.00

error

**Text with highlighted words**

He **don't** like coffee. His friends **drinks** it every day, but **he** never tries. Maybe **he are** **allergic** to caffeine.

## Рисунок Б.7 – Пояснення граматичних помилок від LIME

**'are' (importance score: 0.0085)**

The word "are" is used incorrectly in the phrase "he are allergic" because "he" is singular, and the correct verb to use here is "is." In the corrected version, "Maybe he are allergic" becomes "Maybe he's sensitive," fixing the grammatical mismatch. This error is a classic subject-verb agreement problem, where the subject and verb need to match in number.

**'don' (importance score: 0.0083)**

In the original text, "He don't like coffee" is incorrect because "don't" (a contraction of "do not") is only used with plural subjects or "you." For singular subjects like "he," the correct contraction is "doesn't" (short for "does not"). The correction, "He doesn't like coffee," fixes this error.

**'drinks' (importance score: 0.0083)**

The original sentence says "His friends drinks it every day," but this is incorrect because "His friends" is plural, and with plural subjects, we do not add an "s" to the verb. The correct version, "His friends drink it every day," uses the base form of the verb without "s." This is another subject-verb agreement issue.

Рисунок Б.8 – Інтерпретація пояснення граматичних помилок природною МОВОЮ

Enter the text you want to check for mistakes and press the arrow.  
To get the corrections explained, press Explain button.  
Please note: both correction and explanation take some time, as ML models are working.

A pretty huge bunch of experiments was conducted to evaluate the performance of the model. The results were generally top-notch, although in certain cases the system behaved unpredictably. Further improvements are planned in subsequent iterations.

A large number of experiments were conducted to evaluate the performance of the model. The results were generally excellent, although in some cases the system acted unpredictably. Further improvements are planned for future iterations.

→

**Explain**

Prediction probabilities

correct	0.47
error	0.53

correct      error

the	0.01
subsequent	0.01
certain	0.00
was	0.00
top	0.00
pretty	0.00

**Text with highlighted words**

A pretty huge bunch of experiments was conducted to evaluate the performance of the model. The results were generally top-notch, although in certain cases the system behaved unpredictably. Further improvements are planned in subsequent iterations.

Рисунок Б.9 – Пояснення стилістичних помилок від LIME

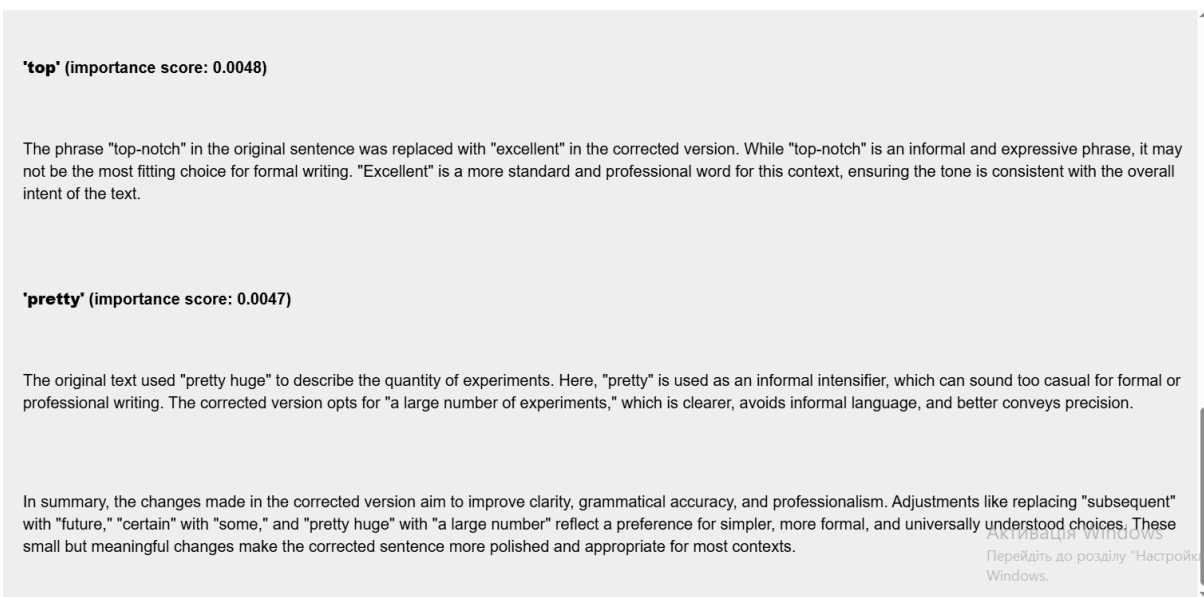


Рисунок Б.10 – Пояснення стилістичних помилок природною мовою

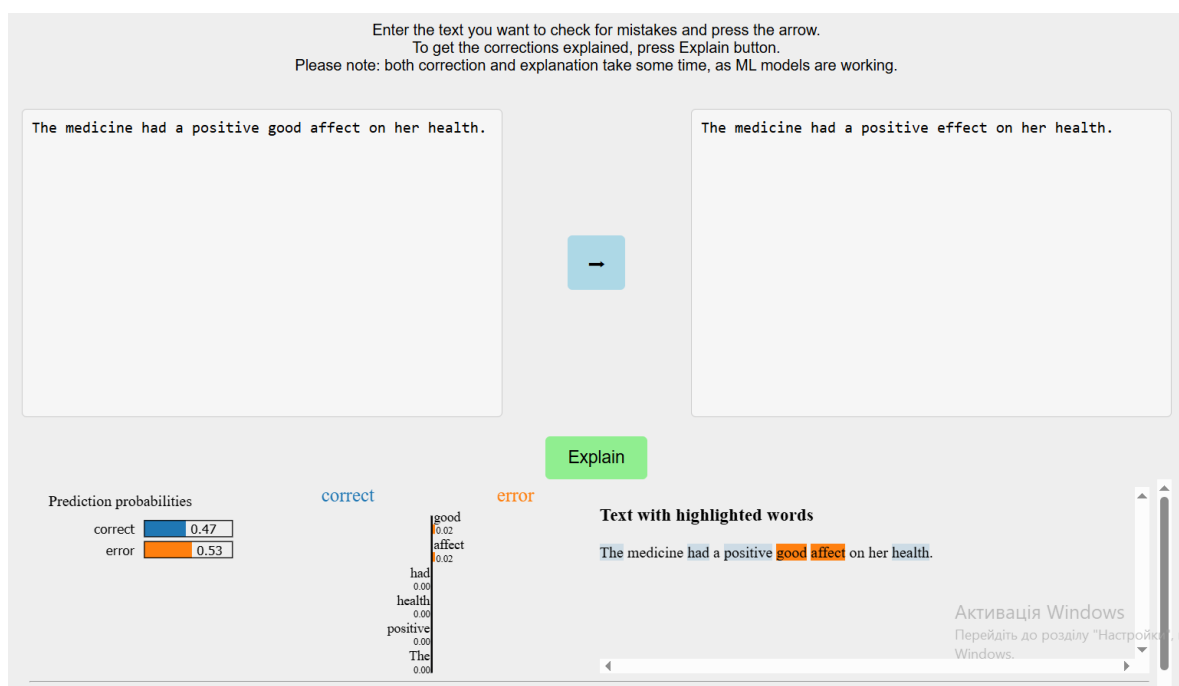


Рисунок Б.11 – Пояснення лексичних помилок від LIME

Let's take a look at the sentence the machine learning model analyzed. The sentence is: "**The medicine had a positive good affect on her health.**". The corrected version is: "**The medicine had a positive effect on her health.**". I'll explain why the correction was made and how each word, as highlighted by the model, contributes to the meaning and correctness of the sentence.

#### 'good'

The word 'good' in the original sentence is unnecessary because 'positive' already conveys the same idea. In English, it is redundant to use multiple words with similar meanings unless you are emphasizing or making comparisons. In this case, removing 'good' makes the sentence clearer and avoids repetition.

#### 'affect'

The word 'affect' in the original sentence is incorrect because it is being used as a noun here. The word 'affect' is most commonly used as a verb, which describes an action (e.g., "The weather affects my mood"). The correct noun for influence or outcome is 'effect,' which is why it is used in the corrected sentence.

Активация Windows

Перейдіть до розділу "Настройки", щ

Рисунок Б.12 – Пояснення лексичних помилок природною мовою

Enter the text you want to check for mistakes and press the arrow.  
To get the corrections explained, press Explain button.  
Please note: both correction and explanation take some time, as ML models are working.

The cat was barking loudly.

The cat was meowing loudly.

→

Explain

Prediction probabilities

correct	0.46
error	0.54

correct

error

barking	0.04
cat	0.00
loudly	0.00
was	0.00
The	0.00

**Text with highlighted words**

The cat was barking loudly.

Активация Windows

Перейдіть до розділу "Настройки", щ

Рисунок Б.13 – Пояснення семантичних помилок від LIME

### Explanation of Important Tokens

**'barking'**: This word is crucial because it describes the sound being made. However, barking is a sound typically associated with dogs, not cats. The model likely flagged this as problematic because the verb does not match the noun "cat." In English, it's important to use verbs that align with the subject, and "barking" is not a logical action for a cat.

**'cat'**: While "cat" itself is not incorrect, it strongly influences the model's decision because it sets the expectation for the sentence. A cat makes specific sounds, such as meowing or purring, and does not bark. This expectation helps us realize the mismatch between the subject and the verb in the original sentence.

**'loudly'**: This word describes how the sound is made. It is not inherently problematic because animals can make loud noises. However, its importance here relates to highlighting the action ("barking") that becomes inconsistent when we consider the subject ("cat").

## Рисунок Б.14 – Пояснення семантичних помилок природною мовою

Enter the text you want to check for mistakes and press the arrow.  
To get the corrections explained, press Explain button.  
Please note: both correction and explanation take some time, as ML models are working.

Hey, the results of the test wasn't really good and a lot of stuff was messed up. The experiment effected the machine badly, so we gotta do it again. Also, the datas shows that some parts don't work proper, which is kinda weird.

→

Hey, the results of the test weren't great, and a lot of things got messed up. The experiment affected the machine badly, so we have to do it again. Also, the data indicates that some parts aren't functioning properly, which is a bit unusual.

Explain

Prediction probabilities

correct	0.46
error	0.54

correct

error

Text with highlighted words

Hey, **the** results of **the** test wasn't really **good** and a lot of stuff was messed up. The experiment effected **the** machine badly, so we gotta do it again. Also, **the** datas shows that some parts **don't** **work** **proper**, which is **kinda** weird.

## Рисунок Б.15 – Пояснення комбінованих помилок від LIME

**Explain**

**'proper'**

The word "proper" is flagged because it is used incorrectly in the phrase "don't work proper" in the original text. In English, we use adverbs to modify verbs, and "proper" is an adjective, not an adverb. The correction fixes this by replacing "work proper" with "functioning properly," where "properly" is the correct adverbial form. This ensures proper grammar and eliminates the informal tone.

**'good'**

The word "good" in the original text is part of the phrase "wasn't really good." The model likely flagged this because "good" is a subjective and general word that doesn't precisely describe the situation. In the corrected sentence, "wasn't really good" is improved to "weren't great." The change also replaces the singular "wasn't" with plural "weren't" for subject-verb agreement. The word "great" is concise and appropriate, enhancing clarity and tone.

**'kinda'**

Finally, "kinda" is highlighted because it is an informal shortening of "kind of," which is not suitable for most formal or professional contexts. The original phrase "which is kinda weird" becomes "which is a bit unusual" in the corrected version. This change eliminates the informal tone and introduces more precise language with "a bit unusual," which is clearer and fits a more formal style.

Активация Windows

Рисунок Б.16 – Пояснення комбінованих помилок природною мовою

