GPT Corrector

```
# Load the necessary assemblies for Windows Forms
Add-Type -AssemblyName System.Windows.Forms

# Add required .NET types for window manipulation
Add-Type @"
    using System;
    using System.Runtime.InteropServices;
    public class Win32 {
        [DllImport("user32.dll")]
        [return: MarshalAs(UnmanagedType.Bool)]
        public static extern bool SetWindowPos(IntPtr hWnd, IntPtr hWndInsertAfter, int X, int
Y, int cx, int cy, uint uFlags);

        public static readonly IntPtr HWND_TOPMOST = new IntPtr(-1);
        public static readonly IntPtr HWND_NOTOPMOST = new IntPtr(-2);
        public const int SWP_NOSIZE = 0x0001;
        public const int SWP_NOMOVE = 0x0002;
        public const int TOPMOST_FLAGS = SWP_NOMOVE | SWP_NOSIZE;
    }
"@

# Define API URL, API Key, System Prompt, and Debug mode as variables
$apiUrl = "https://api.openai.com/v1/chat/completions"
$apiKey = "sk-"
$systemPrompt = "Tu es un correcteur orthographique. Ta seule tâche est de toujours corriger
le texte que tu reçois en entrée."
$debug = $false  # Set to $true to enable debug mode

# Create a form
$form = New-Object System.Windows.Forms.Form
$form.Text = "GPT Grammar Corrector (GGC)"
$form.Size = New-Object System.Drawing.Size(400, 320)
$form.FormBorderStyle = [System.Windows.Forms.FormBorderStyle]::FixedDialog  # Rendre la
fenêtre non redimensionnable
$form.MaximizeBox = $false  # Désactiver le bouton de maximisation
```

```
$form.MinimizeBox = $true  # Vous pouvez laisser le bouton de minimisation si souhaité, sinon mettez à $false
$form.StartPosition = [System.Windows.Forms.FormStartPosition]::CenterScreen  # Centrer la fenêtre à l'écran



# Set form to TopMost
$form.Add_Shown({
    # Obtain the handle of the form and set it to be always on top
    $handle = $form.Handle
    [Win32]::SetWindowPos($handle, [Win32]::HWND_TOPMOST, 0, 0, 0, 0, [Win32]::TOPMOST_FLAGS)
})

# Create a TextBox for input
$inputTextBox = New-Object System.Windows.Forms.TextBox
$inputTextBox.Location = New-Object System.Drawing.Point(10, 10)
$inputTextBox.Size = New-Object System.Drawing.Size(374, 100)
$inputTextBox.Multiline = $true
$inputTextBox.ScrollBars = "Vertical"
$form.Controls.Add($inputTextBox)

# Create a TextBox for output
$outputTextBox = New-Object System.Windows.Forms.TextBox
$outputTextBox.Location = New-Object System.Drawing.Point(11, 120)
$outputTextBox.Size = New-Object System.Drawing.Size(374, 110)
$outputTextBox.Multiline = $true
$outputTextBox.ReadOnly = $true
$outputTextBox.ScrollBars = "Vertical" # Ajout de barres de défilement si nécessaire
$outputTextBox.WordWrap = $true # S'assure que le texte se termine à la ligne suivante si trop long
$form.Controls.Add($outputTextBox)

# Create a Button to submit input
$submitButton = New-Object System.Windows.Forms.Button
$submitButton.Location = New-Object System.Drawing.Point(10, 240)
$submitButton.Size = New-Object System.Drawing.Size(180, 40)
$submitButton.Text = "Envoyer"
$form.Controls.Add($submitButton)

# Create a Button to copy output
```

```
$copyButton = New-Object System.Windows.Forms.Button
$copyButton.Location = New-Object System.Drawing.Point(206, 240)
$copyButton.Size = New-Object System.Drawing.Size(180, 40)
$copyButton.Text = "Copier"
$form.Controls.Add($copyButton)

# Event handler for copy button click
$copyButton.Add_Click({
    [System.Windows.Forms.Clipboard]::SetText($outputTextBox.Text)
})

# Event handler for button click
$submitButton.Add_Click({
    $inputText = [System.Net.WebUtility]::UrlEncode($inputTextBox.Text)

    $headers = @{
        "Authorization" = "Bearer $apiKey"
        "Content-Type"  = "application/json"
    }

    $body = @{
        model = "gpt-4o-mini"
        messages = @(
            @{
                role = "system"
                content = [System.Net.WebUtility]::UrlDecode($systemPrompt)
            }
            @{
                role = "user"
                content = [System.Net.WebUtility]::UrlDecode($inputText) # Ensure original
unicode content
            }
        )
        temperature = 0.7
    } | ConvertTo-Json

    try {
        $response = Invoke-RestMethod -Uri $apiUrl -Method POST -Headers $headers -Body $body
-ContentType "application/json"
        # Directly use the content without additional encoding/decoding
```

```
        $generatedText = $response.choices[0].message.content.Trim()
        $outputTextBox.Text = $generatedText
    } catch {
        if ($debug) {
            $outputTextBox.Text = "Erreur: " + $_.Exception.Message + "`nRequest Body: " +
$body
        } else {
            $outputTextBox.Text = "Erreur: " + $_.Exception.Message
        }
    }
})

# Run the form
[void]$form.ShowDialog()
```

Revision #1
Created 2024-12-04 14:05:49 UTC by Yann Solliard
Updated 2024-12-04 14:06:17 UTC by Yann Solliard