Scripts Powershell

- Send with Sonic SMTP
- Calculateur Salaire
- Download folder clean up
- GPT 2 English
- GPT 2 German
- GPT Corrector

Send with Sonic SMTP

```
# Define the SMTP server details
$smtpServer = "smtp.labnet.local"
$smtpPort = 25

# Define the email details
$to = "yann.solliard@sonicsuisse.ch"
$from = "yanntest@labnet.local"  # Replace with your email address
$subject = "Test Email"
$body = "This is a test email sent via PowerShell."

# Create the SMTP client
$smtp = New-Object System.Net.Mail.SmtpClient($smtpServer, $smtpPort)

# Set the SMTP client credentials if required
# $smtp.Credentials = New-Object System.Net.NetworkCredential("username", "password")

# Create the mail message
$mail = New-Object System.Net.Mail.MailMessage
$mail.From = $from
$mail.To.Add($to)
$mail.Subject = $subject
$mail.Body = $body

# Send the email
try {
    $smtp.Send($mail)
    Write-Host "Email sent successfully."
} catch {
    Write-Error "An error occurred: $_"
}

# Clean up
$mail.Dispose()
```

Calculateur Salaire

```
Add-Type -AssemblyName System.Windows.Forms

# Fonction pour calculer les salaires
function Calculate-Salaries {
    $annualSalary = [double]$txtAnnualSalary.Text
    $hoursPerWeek = [double]$txtHoursPerWeek.Text
    $percentage = [double]$txtPercentage.Text / 100
    $salaryDivision = [double]$cboSalaryDivision.SelectedItem

    if ($hoursPerWeek -eq 0) {
        [System.Windows.Forms.MessageBox]::Show("Le nombre d'heures par semaine ne peut pas être zéro.", "Erreur", [System.Windows.Forms.MessageBoxButtons]::OK, [System.Windows.Forms.MessageBoxIcon]::Error)
        return
    }

    if ($percentage -lt 0 -or $percentage -gt 1) {
        [System.Windows.Forms.MessageBox]::Show("Le pourcentage doit être entre 0 et 100.", "Erreur", [System.Windows.Forms.MessageBoxButtons]::OK, [System.Windows.Forms.MessageBoxIcon]::Error)
        return
    }

    $adjustedAnnualSalary = $annualSalary * $percentage
    $hourlyRate = $adjustedAnnualSalary / (52 * $hoursPerWeek)
    $dailyRate = $hourlyRate * ($hoursPerWeek / 5)  # Supposition de 5 jours de travail par semaine
    $weeklyRate = $hourlyRate * $hoursPerWeek
    $monthlyRate = $adjustedAnnualSalary / $salaryDivision

    $lblHourlyRate.Text = "Salaire Horaire : CHF " + [math]::Round($hourlyRate, 2)
    $lblDailyRate.Text = "Salaire Journalier : CHF " + [math]::Round($dailyRate, 2)
    $lblWeeklyRate.Text = "Salaire Hebdomadaire : CHF " + [math]::Round($weeklyRate, 2)
    $lblMonthlyRate.Text = "Salaire Mensuel : CHF " + [math]::Round($monthlyRate, 2)
}
```

```
# Créer la fenêtre
$form = New-Object System.Windows.Forms.Form
$form.Text = "Calculateur de Salaire"
$form.Size = New-Object System.Drawing.Size(320,380)
$form.StartPosition = "CenterScreen"

# Label et textbox pour le salaire annuel
$lblAnnualSalary = New-Object System.Windows.Forms.Label
$lblAnnualSalary.Text = "Salaire Annuel (CHF):"
$lblAnnualSalary.AutoSize = $true
$lblAnnualSalary.Location = New-Object System.Drawing.Point(10,20)

$txtAnnualSalary = New-Object System.Windows.Forms.TextBox
$txtAnnualSalary.Location = New-Object System.Drawing.Point(160,17)
$txtAnnualSalary.Width = 120

# Label et textbox pour le nombre d'heures par semaine
$lblHoursPerWeek = New-Object System.Windows.Forms.Label
$lblHoursPerWeek.Text = "Heures par Semaine:"
$lblHoursPerWeek.AutoSize = $true
$lblHoursPerWeek.Location = New-Object System.Drawing.Point(10,60)

$txtHoursPerWeek = New-Object System.Windows.Forms.TextBox
$txtHoursPerWeek.Location = New-Object System.Drawing.Point(160,57)
$txtHoursPerWeek.Width = 120

# Label et textbox pour le pourcentage
$lblPercentage = New-Object System.Windows.Forms.Label
$lblPercentage.Text = "Pourcentage de Travail (%):"
$lblPercentage.AutoSize = $true
$lblPercentage.Location = New-Object System.Drawing.Point(10,100)

$txtPercentage = New-Object System.Windows.Forms.TextBox
$txtPercentage.Location = New-Object System.Drawing.Point(160,97)
$txtPercentage.Width = 120

# Combobox pour choisir 12 ou 13 salaires
$lblSalaryDivision = New-Object System.Windows.Forms.Label
$lblSalaryDivision.Text = "Divisions Annuelles :"
```

```
$lblSalaryDivision.AutoSize = $true
$lblSalaryDivision.Location = New-Object System.Drawing.Point(10,140)

$cboSalaryDivision = New-Object System.Windows.Forms.ComboBox
$cboSalaryDivision.Items.AddRange(@(12, 13))
$cboSalaryDivision.SelectedIndex = 0  # Par défaut 12
$cboSalaryDivision.Location = New-Object System.Drawing.Point(160,137)
$cboSalaryDivision.Width = 120

# Bouton pour calculer
$btnCalculate = New-Object System.Windows.Forms.Button
$btnCalculate.Text = "Calculer"
$btnCalculate.Location = New-Object System.Drawing.Point(100,180)
$btnCalculate.Add_Click({Calculate-Salaries})

# Labels pour afficher les résultats
$lblHourlyRate = New-Object System.Windows.Forms.Label
$lblHourlyRate.AutoSize = $true
$lblHourlyRate.Location = New-Object System.Drawing.Point(10,220)

$lblDailyRate = New-Object System.Windows.Forms.Label
$lblDailyRate.AutoSize = $true
$lblDailyRate.Location = New-Object System.Drawing.Point(10,250)

$lblWeeklyRate = New-Object System.Windows.Forms.Label
$lblWeeklyRate.AutoSize = $true
$lblWeeklyRate.Location = New-Object System.Drawing.Point(10,280)

$lblMonthlyRate = New-Object System.Windows.Forms.Label
$lblMonthlyRate.AutoSize = $true
$lblMonthlyRate.Location = New-Object System.Drawing.Point(10,310)

# Ajouter tous les contrôles à la fenêtre
$form.Controls.Add($lblAnnualSalary)
$form.Controls.Add($txtAnnualSalary)
$form.Controls.Add($lblHoursPerWeek)
$form.Controls.Add($txtHoursPerWeek)
$form.Controls.Add($lblPercentage)
$form.Controls.Add($txtPercentage)
$form.Controls.Add($lblSalaryDivision)
```

```
$form.Controls.Add($cboSalaryDivision)
$form.Controls.Add($btnCalculate)
$form.Controls.Add($lblHourlyRate)
$form.Controls.Add($lblDailyRate)
$form.Controls.Add($lblWeeklyRate)
$form.Controls.Add($lblMonthlyRate)

# Afficher la fenêtre
$form.Add_Shown({$form.Activate()})
[void]$form.ShowDialog()
```

Download folder clean up

```
# Import the necessary namespace to work with the file system.
Add-Type -AssemblyName Microsoft.VisualBasic

# Set the source directory.
$sourcePath = "\\usr-cos\homeSTD$\ysolliard\Downloads"

# Get the current date and subtract 5 days to find the cutoff date.
$cutOffDate = (Get-Date).AddDays(-5)

# Get all files and directories in the source path.
$items = Get-ChildItem -Path $sourcePath -Recurse

# Iterate over each item.
foreach ($item in $items) {
    # Check if the item's LastWriteTime is older than the cutoff date.
    if ($item.LastWriteTime -lt $cutOffDate) {
        try {
            if ($item.PSIsContainer) {
                # If the item is a directory, move it to the Recycle Bin.
                [Microsoft.VisualBasic.FileIO.FileSystem]::DeleteDirectory(
                    $item.FullName,
                    [Microsoft.VisualBasic.FileIO.UIOption]::OnlyErrorDialogs,
                    [Microsoft.VisualBasic.FileIO.RecycleOption]::SendToRecycleBin
                )
                Write-Host "Directory '$($item.FullName)' moved to Recycle Bin."
            } else {
                # If the item is a file, move it to the Recycle Bin.
                [Microsoft.VisualBasic.FileIO.FileSystem]::DeleteFile(
                    $item.FullName,
                    [Microsoft.VisualBasic.FileIO.UIOption]::OnlyErrorDialogs,
                    [Microsoft.VisualBasic.FileIO.RecycleOption]::SendToRecycleBin
                )
                Write-Host "File '$($item.FullName)' moved to Recycle Bin."
            }
        }
```

```
        catch {
            Write-Host "Error moving item '$($item.FullName)' to Recycle Bin:
$($_.Exception.Message)"
        }
    }
}
```

GPT 2 English

```
# 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 = "Translate everything to english and nothing else."
$debug = $false  # Set to $true to enable debug mode

# Create a form
$form = New-Object System.Windows.Forms.Form
$form.Text = "GPT 2 Enlish (G2E)"
$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()
```

GPT 2 German

```
# 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 = "Translate everything to german and nothing else. Never use ß replace with ss
instead."
$debug = $false  # Set to $true to enable debug mode

# Create a form
$form = New-Object System.Windows.Forms.Form
$form.Text = "GPT 2 German (G2G)"
$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 = 1
    } | 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()
```

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()
```