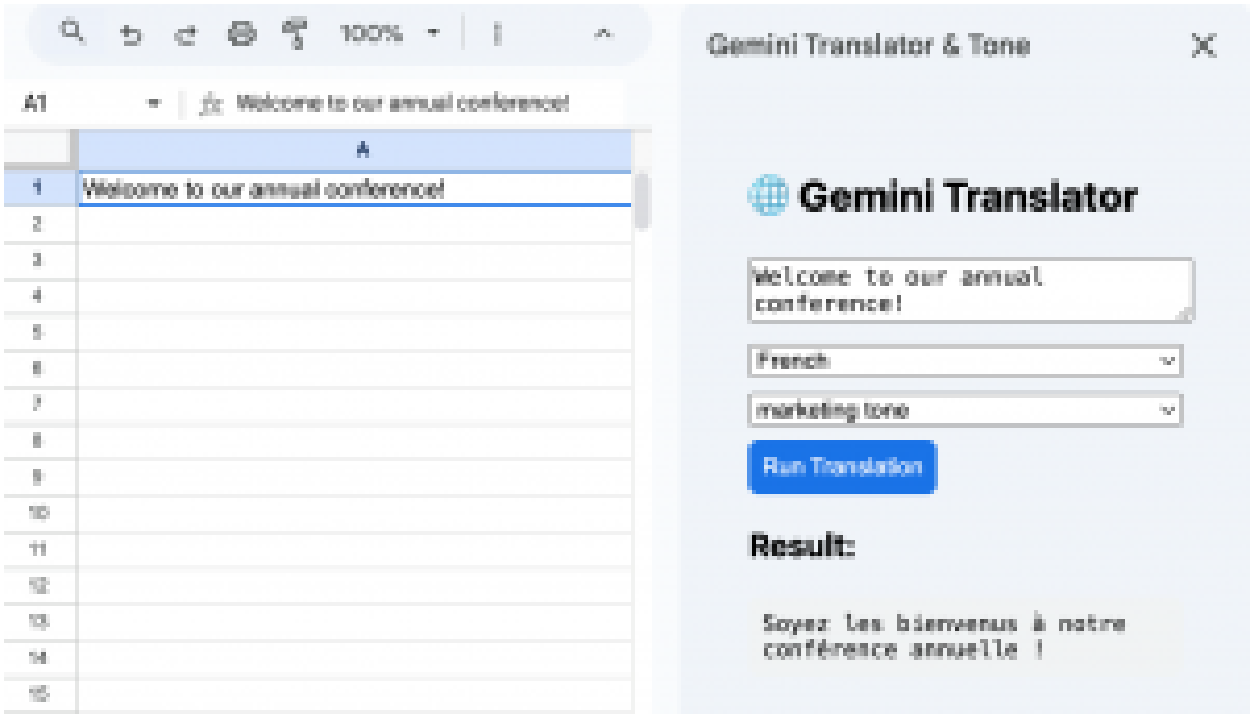
# 🌍 Build an AI Translator & Tone Converter in Google Sheets (with Gemini & Apps Script)




## 🚀 Overview

In this exercise, you'll learn how to connect **Google Sheets** to **Gemini's REST API** using **Apps Script**, enabling you to automatically translate text and rewrite it in different tones — all directly from your spreadsheet.

You'll build:

- A **custom menu** with easy one-click translation and tone tools
- A **sidebar interface** for interactive translation
- **Custom functions** like `=AI_TRANSLATE(A2, "French")` and `=AI_TONE(A2, "friendly tone")`
- A **robust Gemini integration** that handles long text, token limits, and model safety

---

## 🧠 Learning Objectives

By completing this exercise, you'll learn how to:

1. Use `UrlFetchApp` to call the **Gemini REST API** from Apps Script.
2. Implement **chunking logic** to process long text safely.
3. Extract model responses and handle truncation or safety blocks gracefully.
4. Add interactive UI elements with Apps Script's **Sidebar service**.
5. Build reusable, production-ready AI utilities in Google Sheets.

---

## 🧩 Step 1 — Setup the Project

1. Open a new **Google Sheet**.
2. Go to **Extensions** → **Apps Script**.
3. Rename your project to:
   **Gemini Translator & Tone Converter**
4. Create three files:
   - `Code.gs`
   - `Sidebar.html`
   - `appsscript.json`

---

## 🧾 Step 2 — Paste the Updated Code.gs

Here's the **fully working script**, with token headroom, chunking, and language normalization:

```
/**
 * Exercise 4 – AI Translator & Tone Converter for Google Sheets
```

```
(Gemini REST)
 * Author: Laurence “Lars” Svekis
 *
 * Features:
 *  - Translate selection → next column
 *  - Tone-convert selection → next column
 *  - Sidebar UI: translate & tone settings (see Sidebar.html)
 *  - Custom functions: =AI_TRANSLATE(), =AI_TONE()
 */


const GEMINI_API_KEY = ''; // Optional fallback, prefer Script
Properties
const GEMINI_MODEL   = 'gemini-2.5-flash';
const BASE_URL       =
'https://generativelanguage.googleapis.com/v1beta';


const DEFAULT_TEMP   = 0.3;
const DEFAULT_TOKENS = 1024;
const MAX_TOKENS_HARD_CAP = 2048;
const COOLDOWN_MS    = 120;


const CHUNK_CHAR_LIMIT = 3000;
const CHUNK_JOINER     = '\n';


// === MENU ===
function onOpen() {
```

```
  SpreadsheetApp.getUi()
    .createMenu('🧠 AI Tools')
    .addItem('Open Translator Sidebar', 'showTranslatorSidebar')
    .addSeparator()
    .addItem('Translate Selection → next column',
'translateSelectionToRight')
    .addItem('Tone-convert Selection → next column',
'toneSelectionToRight')
    .addSeparator()
    .addItem('Insert Sample Data', 'insertSampleData')
    .addToUi();
}

function showTranslatorSidebar() {
  const html = HtmlService.createHtmlOutputFromFile('Sidebar')
    .setTitle('Gemini Translator & Tone');
  SpreadsheetApp.getUi().showSidebar(html);
}

// === API KEY ===
function getApiKey() {
  const prop =
PropertiesService.getScriptProperties().getProperty('GEMINI_API_KEY');
  return prop && prop.trim() ? prop.trim() : GEMINI_API_KEY;
}
```

```
// === ROBUST RESPONSE EXTRACTION ===

function extractGeminiText(json) {

  try {

    if (json?.error?.message) return `❌ API error:
${json.error.message}`;



    const block = json?.promptFeedback?.blockReason;

    if (block) {

      const details = (json?.promptFeedback?.safetyRatings || [])

        .map(r => `${r.category}:${r.probability}`).join(', ');

      return `❌ Blocked by safety (${block}${details ? ' — ' +
details : ''}).`;

    }



    const cand = json?.candidates?.[0];

    if (!cand) return '';



    if (cand.finishReason === 'MAX_TOKENS') {

      const parts = cand?.content?.parts || [];

      const partial = parts.map(p => p?.text || '').join('').trim();

      return partial

        ? `${partial}\n\n⚠️ Truncated (MAX_TOKENS). Increase max
tokens or shorten input.`

        : '⚠️ Truncated (MAX_TOKENS) and no visible text.';

    }
```

```
    const parts = cand?.content?.parts || [];
    const text = parts.map(p => p?.text || '').join('').trim();
    if (text) return text;

    const cText = cand?.text;
    if (cText && String(cText).trim()) return String(cText).trim();

    return '';
  } catch (e) {
    return '';
  }
}

// === GENERATE TEXT (Gemini Request) ===
function geminiGenerateText(prompt, { temperature = DEFAULT_TEMP,
maxTokens = DEFAULT_TOKENS } = {}) {
  const apiKey = getApiKey();
  if (!apiKey) return '❌ Set GEMINI_API_KEY first in Script
Properties.';

  const capped = Math.min(Number(maxTokens) || DEFAULT_TOKENS,
MAX_TOKENS_HARD_CAP);
  const payload = {
    contents: [{ role: "user", parts: [{ text: prompt }] }],
    generationConfig: { temperature, maxOutputTokens: capped, topP:
0.95, topK: 40 }
  };
```

```
  const url =
`${BASE_URL}/models/${encodeURIComponent(GEMINI_MODEL)}:generateConten
t?key=${encodeURIComponent(apiKey)}`;

  const options = { method: 'post', contentType: 'application/json',
payload: JSON.stringify(payload), muteHttpExceptions: true };


  try {

    const res  = UrlFetchApp.fetch(url, options);

    const code = res.getResponseCode();

    const body = res.getContentText();


    if (code < 200 || code >= 300) return `❌ Error: Gemini HTTP
${code}. ${body}`;

    const json = JSON.parse(body);

    const text = extractGeminiText(json);

    return text || '⚠️ No output (see Logs).';

  } catch (e) {

    return '❌ Error: ' + e.message;

  }

}


// === CHUNKING ===

function splitTextIntoChunks(input, limit = CHUNK_CHAR_LIMIT) {

  const text = String(input || '').replace(/\r\n/g, '\n');

  if (text.length <= limit) return [text];
```

```
  const paras = text.split(/\n{2,}/);
  if (paras.length > 1) return packChunks_(paras, limit);

  const sents = text.split(/(?<=[.!?])\s+/);
  if (sents.length > 1) return packChunks_(sents, limit);

  const chunks = [];
  for (let i = 0; i < text.length; i += limit)
chunks.push(text.slice(i, i + limit));
  return chunks;
}

function packChunks_(parts, limit) {
  const out = [];
  let buf = '';
  for (const part of parts) {
    const candidate = buf ? buf + '\n\n' + part : part;
    if (candidate.length > limit) {
      if (buf) out.push(buf);
      if (part.length > limit) {
        for (let i = 0; i < part.length; i += limit)
out.push(part.slice(i, i + limit));
        buf = '';
      } else buf = part;
    } else buf = candidate;
  }
```

```
  if (buf) out.push(buf);
  return out;
}

// === LANGUAGE NORMALIZATION ===
function normalizeLanguage(lang) {
  if (!lang) return 'English';
  const s = String(lang).trim().toLowerCase();
  const map = {
    en:'English', fr:'French', es:'Spanish', de:'German',
    it:'Italian', pt:'Portuguese', hi:'Hindi', ja:'Japanese',
    ko:'Korean', zh:'Chinese'
  };
  return map[s] || lang;
}

// === TRANSLATE & TONE ===
function callGeminiTranslate(text, targetLanguage, tone, temperature,
maxTokens) {
  const tgt = normalizeLanguage(targetLanguage || 'English');
  const t   = tone ? ` and adjust tone to "${tone}"` : '';
  const mkPrompt = (chunk) => [
    `Translate the text below into ${tgt}${t}.`,
    'Keep meaning accurate; do not add or drop information.',
    'Return ONLY the translation as plain text.',
```

```
    '',
    chunk
  ].join('\n');

  const chunks = splitTextIntoChunks(text);
  const outs = [];
  for (const c of chunks) {
    const out = geminiGenerateText(mkPrompt(c), {
      temperature, maxTokens: maxTokens || DEFAULT_TOKENS * 2
    });
    outs.push(out);
    Utilities.sleep(COOLDOWN_MS);
  }
  return outs.join(CHUNK_JOINER);
}

function callGeminiTone(text, tone, temperature, maxTokens) {
  const tn = tone || 'professional tone';
  const mkPrompt = (chunk) => [
    `Rewrite the text below in a ${tn}.`,
    'Preserve meaning; return ONLY the rewritten text.',
    '',
    chunk
  ].join('\n');
```

```
  const chunks = splitTextIntoChunks(text);
  const outs = [];
  for (const c of chunks) {
    const out = geminiGenerateText(mkPrompt(c), {
      temperature, maxTokens: maxTokens || DEFAULT_TOKENS * 2
    });
    outs.push(out);
    Utilities.sleep(COOLDOWN_MS);
  }
  return outs.join(CHUNK_JOINER);
}

// === CUSTOM FUNCTIONS ===
function AI_TRANSLATE(text, lang, tone) {
  return callGeminiTranslate(text, lang, tone, DEFAULT_TEMP,
DEFAULT_TOKENS);
}

function AI_TONE(text, tone) {
  return callGeminiTone(text, tone, DEFAULT_TEMP, DEFAULT_TOKENS);
}

// === SHEETS ACTIONS ===
function translateSelectionToRight() {
  const range = SpreadsheetApp.getActiveRange();
```

```
  const ui = SpreadsheetApp.getUi();

  const resp = ui.prompt('Translate Selection', 'Enter target
language:', ui.ButtonSet.OK_CANCEL);

  if (resp.getSelectedButton() !== ui.Button.OK) return;

  const lang = resp.getResponseText().trim() || 'English';



  const values = range.getValues();

  const out = values.map(r => [callGeminiTranslate(r[0], lang)]);

  range.offset(0, 1).setValues(out);

  SpreadsheetApp.getActive().toast(`Translated ${values.length}
row(s)`);

}



function toneSelectionToRight() {

  const range = SpreadsheetApp.getActiveRange();

  const ui = SpreadsheetApp.getUi();

  const resp = ui.prompt('Tone Conversion', 'Enter desired tone:',
ui.ButtonSet.OK_CANCEL);

  if (resp.getSelectedButton() !== ui.Button.OK) return;

  const tone = resp.getResponseText().trim() || 'professional tone';



  const values = range.getValues();

  const out = values.map(r => [callGeminiTone(r[0], tone)]);

  range.offset(0, 1).setValues(out);

  SpreadsheetApp.getActive().toast(`Tone-converted ${values.length}
row(s)`);

}
```

```
// === SAMPLE DATA ===
function insertSampleData() {
  const sheet = SpreadsheetApp.getActiveSheet();
  sheet.clearContents();
  sheet.getRange(1, 1, 6, 2).setValues([
    ['Original Text', 'Result'],
    ['Please submit your report by Friday.', ''],
    ['Welcome to our annual conference!', ''],
    ['The product helps students learn faster.', ''],
    ['Thank you for your feedback!', ''],
    ['Our platform supports over 10 languages.', '']
  ]);
}
```

---

## 🧮 Step 3 — Sidebar.html

It connects to `callGeminiTranslate()` and displays output instantly.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <style>
   body { font-family: system-ui; padding: 16px; }
```

```
    textarea, select, input { width: 100%; margin-bottom: 8px; }
    button { background: #1a73e8; color: white; border: 0; padding: 8px; border-radius: 6px; cursor: pointer; }
    pre { background: #f1f3f4; padding: 8px; border-radius: 6px; white-space: pre-wrap; }
  </style>
 </head>
 <body>
  <h2>🌐 Gemini Translator</h2>
  <textarea id="text" placeholder="Enter text here..."></textarea>
  <select id="lang">
<option>French</option><option>Spanish</option><option>German</option><option>Italian</option>
  </select>
  <select id="tone">
   <option value="">(No tone change)</option>
   <option>friendly tone</option>
   <option>professional tone</option>
   <option>marketing tone</option>
  </select>
  <button onclick="run()">Run Translation</button>
  <h3>Result:</h3>
  <pre id="out">Waiting...</pre>

  <script>
   function run() {
```

```
    const text = document.getElementById('text').value;
    const lang = document.getElementById('lang').value;
    const tone = document.getElementById('tone').value;
    document.getElementById('out').textContent = '⏳ Processing...';
    google.script.run
      .withSuccessHandler(r => document.getElementById('out').textContent = r)
      .callGeminiTranslate(text, lang, tone);
  }
</script>
</body>
</html>
```

---

## 🧩 Step 4 — Understanding the Code

### 1️⃣ `geminiGenerateText()`

This function handles API requests safely:

- Builds the JSON payload.
- Sends it using `UrlFetchApp.fetch`.
- Extracts clean text using `extractGeminiText()`.
- Handles truncation, safety blocks, and errors.

### 2️⃣ `splitTextIntoChunks()`

Large texts can exceed token limits. This function:

- Splits text by paragraph or sentence.
- Packs them into safe ~3K-character chunks.

- Processes each piece sequentially.

### 3️⃣ `callGeminiTranslate()` and `callGeminiTone()`

These handle:

- Prompt generation ("Translate this text into French…")
- Sending each chunk to Gemini.
- Reassembling the full result.

### 4️⃣ Custom Functions

You can use these directly inside Sheets:

```
=AI_TRANSLATE(A2, "Spanish", "friendly tone")
```

```
=AI_TONE(A2, "formal business tone")
```

### 5️⃣ Menu Actions

From the custom 🧠 **AI Tools** menu:

- "Translate Selection → next column"
- "Tone-convert Selection → next column"
- "Insert Sample Data"

### 6️⃣ Sidebar Interface

The sidebar lets you experiment interactively, running translation or tone tasks without typing formulas.

---

## 🧪 Step 5 — Testing

Run these test cases in your Sheet:

| A (Original Text) | B (Output) |
| --- | --- |
| “Welcome to our annual conference!” | → “Bienvenue à notre conférence annuelle !” |
| “Please submit your report by Friday.” | → “Could you please send your report by Friday? Thanks!” |

---

## ⚙️ Step 6 — Authorization Scopes

Add these scopes to your `appsscript.json`:

```
{
  "timeZone": "America/Toronto",
  "exceptionLogging": "STACKDRIVER",
  "runtimeVersion": "V8",
  "oauthScopes": [
    "https://www.googleapis.com/auth/spreadsheets",
    "https://www.googleapis.com/auth/script.container.ui",
    "https://www.googleapis.com/auth/script.external_request"
  ]
}
```

---

## ✅ Step 7 — Key Takeaways

- **Chunking** prevents truncation.
- **MAX_TOKENS_HARD_CAP** ensures responses fit within model limits.
- **extractGeminiText()** gives readable errors instead of blanks.

- **Custom UI + functions** turn Sheets into a real AI assistant.