Text-to-Speech Translation Application on EC2

Rizvan Nahif
Machine Learning Cloud Comp - AIGC-5003-0NA

1. Project Overview

This application allows users to enter text on a web interface, which is then translated to a language (Spanish or French) using Amazon Translate and converted to audio using Amazon Polly. The translated text is played back as audio directly on the web interface. This document provides detailed instructions on setting up and running the application on an Amazon EC2 instance in the ca-central-1 region.

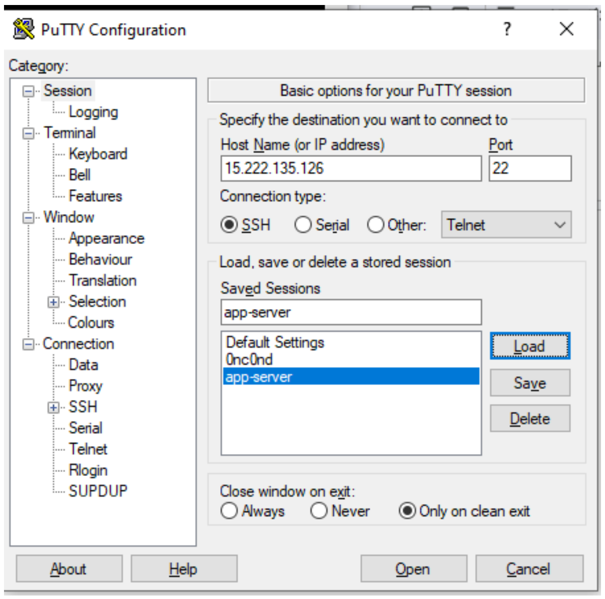
2. Step-by-Step Setup Instructions

1. Navigate to EC2 Dashboard:
   - Go to the AWS Management Console and open the EC2 Dashboard in the ca-central-1 region.
2. Launch EC2 Instance:
   - Select Launch Instance.
   - Choose Amazon Linux 2 AMI as the operating system for compatibility with AWS services.
3. Instance Settings:
   - Select t2.micro (free-tier eligible) as the instance type.
   - In Configure Instance Details, set:
     - Subnet: Choose sbn-fast-ai-academic-1-publicca-central-1a.
     - IAM Role: Assign the IAM role fast-ai-academic-1-Student-EC2.
     - Availability Zone: Ensure the instance is in AZ 1a or 1b.
4. Configure Security Group:
   - Create or assign a security group with inbound rules to allow:
     - SSH (port 22) for server access.
     - HTTP (port 80) for web server access.
     - Custom TCP (Port 8080) for web service access
5. Launch the Instance: Complete the launch and associate the existing Elastic IP (EIP) to the instance to ensure consistent access.

3. Connect to the EC2 Instance Using PuTTY on Windows

1. Download and Install PuTTY:
   - If PuTTY is not installed, download it from the official site here.
2. Download .ppk key pair from EC2 instance:
   - Open PuTTYgen (part of the PuTTY suite).
   - Click Load, and select your .ppk file.

    - Click Save private key and save it as a .ppk file, which PuTTY uses.
3. Launch PuTTY:
    - Open PuTTY and enter the Elastic IP (EIP) of your EC2 instance in the Host Name field.
4. Configure SSH Authentication:
    - In the left menu, navigate to Connection -> SSH -> Auth.
    - Browse to select your .ppk file for authentication.
5. Open Connection:
    - Click Open to start the SSH session. When prompted, accept the security alert.
    - Log in as ec2-user.



Specify the destination you want to connect to
Host Name (or IP address): 15.222.135.126
Port: 22
Connection type: [✓ SSH] Serial Other: Telnet
Load, save or delete a stored session
Saved Sessions: app-server
Default Settings
0nc0nd
app-server
Close window on exit: Always Never [✓ Only on clean exit]

4. Install Required Libraries

1. pip install flask
2. pip install boto3

5. Cloning GitHub Repository on EC2

1. On your EC2 instance, create and navigate to the directory where you want to store the repository files.

    mkdir my_project

    cd my_project

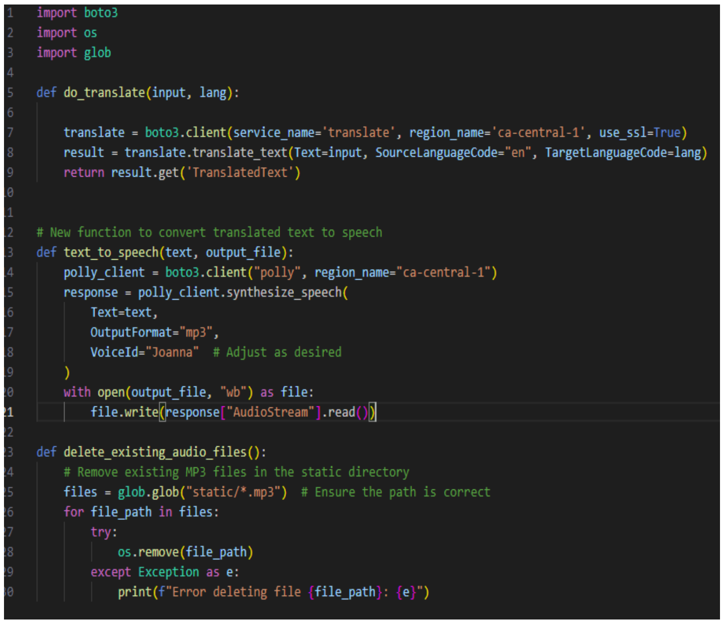
2. Clone the Repository Using HTTPS

   ```
   git clone https://github.com/joyrizvan/amazon-web-app.git
   ```

3. Pull Updates
   ```
   git pull
   ```

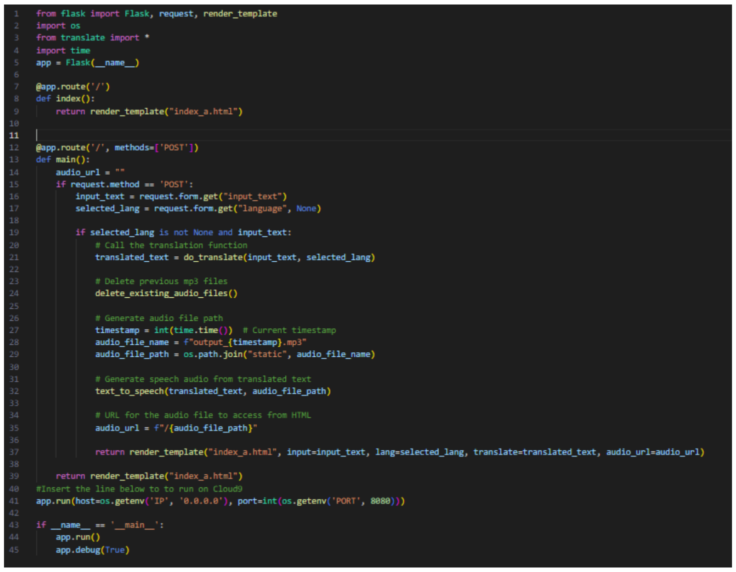
6. Directories and files

1. .py files - There is a template.py file that contains the methods to call the amazon services.



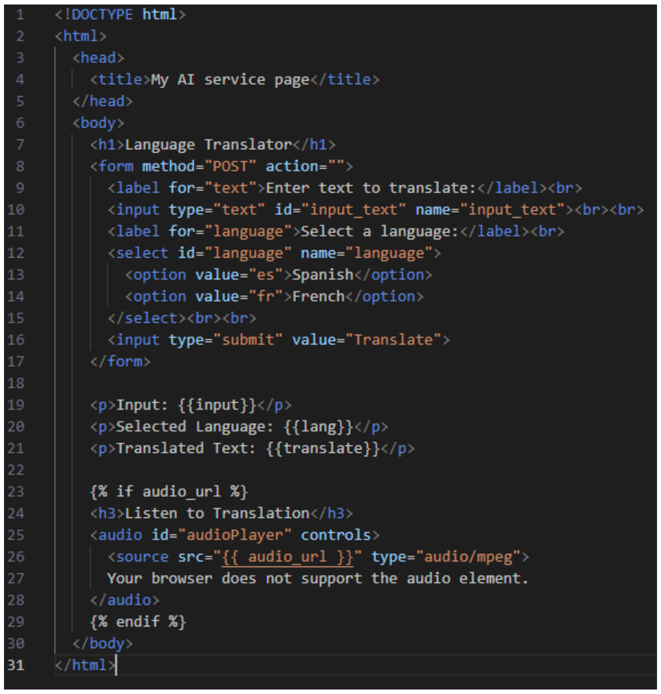
```python
import boto3
import os
import glob

def do_translate(input, lang):

    translate = boto3.client(service_name='translate', region_name='ca-central-1', use_ssl=True)
    result = translate.translate_text(Text=input, SourceLanguageCode="en", TargetLanguageCode=lang)
    return result.get('TranslatedText')


# New function to convert translated text to speech
def text_to_speech(text, output_file):
    polly_client = boto3.client("polly", region_name="ca-central-1")
    response = polly_client.synthesize_speech(
        Text=text,
        OutputFormat="mp3",
        VoiceId="Joanna"  # Adjust as desired
    )
    with open(output_file, "wb") as file:
        file.write(response["AudioStream"].read())

def delete_existing_audio_files():
    # Remove existing MP3 files in the static directory
    files = glob.glob("static/*.mp3")  # Ensure the path is correct
    for file_path in files:
        try:
            os.remove(file_path)
        except Exception as e:
            print(f"Error deleting file {file_path}: {e}")
```

There is a run.py file that will run the application and initialize the main application.



```python
from flask import Flask, request, render_template
import os
from translate import *
import time
app = Flask(__name__)

@app.route('/')
def index():
    return render_template("index_a.html")


@app.route('/', methods=['POST'])
def main():
    audio_url = ""
    if request.method == 'POST':
        input_text = request.form.get("input_text")
        selected_lang = request.form.get("language", None)

        if selected_lang is not None and input_text:
            # Call the translation function
            translated_text = do_translate(input_text, selected_lang)

            # Delete previous mp3 files
            delete_existing_audio_files()

            # Generate audio file path
            timestamp = int(time.time())  # Current timestamp
            audio_file_name = f"output_{timestamp}.mp3"
            audio_file_path = os.path.join("static", audio_file_name)

            # Generate speech audio from translated text
            text_to_speech(translated_text, audio_file_path)

            # URL for the audio file to access from HTML
            audio_url = f"/{audio_file_path}"

            return render_template("index_a.html", input=input_text, lang=selected_lang, translate=translated_text, audio_url=audio_url)

    return render_template("index_a.html")
#Insert the line below to to run on Cloud9
app.run(host=os.getenv('IP', '0.0.0.0'), port=int(os.getenv('PORT', 8080)))

if __name__ == '__main__':
    app.run()
    app.debug(True)
```

2. The template directory contains the HTML file



```html
<!DOCTYPE html>
<html>
  <head>
    <title>My AI service page</title>
  </head>
  <body>
    <h1>Language Translator</h1>
    <form method="POST" action="">
      <label for="text">Enter text to translate:</label><br>
      <input type="text" id="input_text" name="input_text"><br><br>
      <label for="language">Select a language:</label><br>
      <select id="language" name="language">
        <option value="es">Spanish</option>
        <option value="fr">French</option>
      </select><br><br>
      <input type="submit" value="Translate">
    </form>

    <p>Input: {{input}}</p>
    <p>Selected Language: {{lang}}</p>
    <p>Translated Text: {{translate}}</p>

    {% if audio_url %}
    <h3>Listen to Translation</h3>
    <audio id="audioPlayer" controls>
      <source src="{{ audio_url }}" type="audio/mpeg">
      Your browser does not support the audio element.
    </audio>
    {% endif %}
  </body>
</html>
```

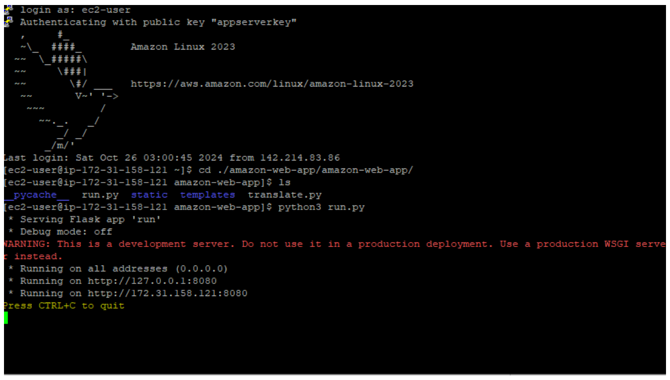
3. The static directory is where the audio from Amazon Polly is stored as .mp3

7. Running the Application

Run run.py

python3 run.py

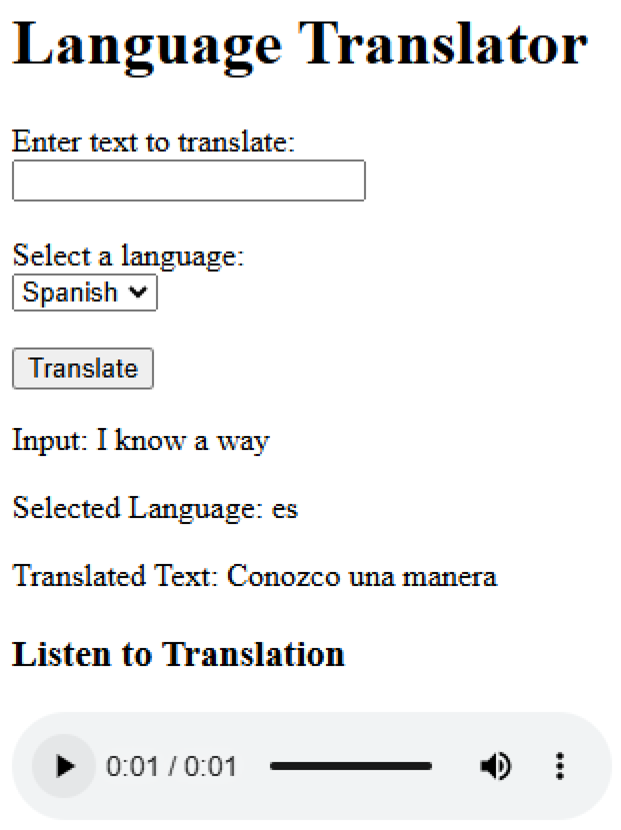
The application uses the default flask wsgi server which is not recommended for production.



```
login as: ec2-user
Authenticating with public key "appserverkey"
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Oct 26 03:00:45 2024 from 142.214.83.86
[ec2-user@ip-172-31-158-121 ~]$ cd ./amazon-web-app/amazon-web-app/
[ec2-user@ip-172-31-158-121 amazon-web-app]$ ls
__pycache__  run.py  static  templates  translate.py
[ec2-user@ip-172-31-158-121 amazon-web-app]$ python3 run.py
 * Serving Flask app 'run'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI serve
r instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://172.31.158.121:8080
Press CTRL+C to quit
```

8. Front End

Use publicip:port to access the webpage.



Input: I know a way

Selected Language: es

Translated Text: Conozco una manera

- Write a sentence in English in the text section
- Select a language (Spanish or French)
- Click on play to listen to the translated audio.

9. Conclusion

The application is a good demonstration of Amazon’s two services Translate and Polly. This small scale translation applications can easily be built using the services.