Translating and speaking text from a photo with glossaries

This page shows how to detect text in an image, how to personalize translations, and how to generate synthetic speech from text. This tutorial uses Cloud Vision to detect text in an image file. Then, this tutorial shows how to use Cloud Translation to provide a custom translation of the detected text. Finally, this tutorial uses Text-to-Speech to provide machine dictation of the translated text.

Objectives

1. Pass text recognized by the Cloud Vision API to the Cloud Translation API.
2. Create and use Cloud Translation glossaries to personalize Cloud Translation API translations.
3. Create an audio representation of translated text using the Text-to-Speech API.

Costs

Each Google Cloud API uses a separate pricing structure.

For pricing details, refer to the Cloud Vision pricing guide (https://cloud.google.com/vision/pricing/?hl=zh-tw), the Cloud Translation pricing guide (https://cloud.google.com/translate/pricing?hl=zh-tw), and the Text-to-Speech pricing guide (https://cloud.google.com/text-to-speech/pricing?hl=zh-tw).

Before you begin

Make sure that you have:

- A project in the Google Cloud Console (https://console.cloud.google.com/?hl=zh-tw) with the Vision API, the Cloud Translation API, and the Text-to-Speech API enabled (https://cloud.google.com/apis/docs/getting-started?hl=zh-tw#enabling_apis)
- A basic familiarity with Python (https://www.python.org/) programming

Downloading the code samples

This tutorial uses code in the **`translate/cloud-client/hybrid_glossaries`** directory of the Google Cloud Python samples (https://github.com/GoogleCloudPlatform/python-docs-samples).

To download and navigate to the code for this tutorial, run the following commands from the terminal.

```
git clone https://github.com/GoogleCloudPlatform/python-docs-samples.g
cd python-docs-samples/translate/cloud-client/hybrid_glossaries/
```

Setting up client libraries

This tutorial uses Vision (https://cloud.google.com/vision/docs/reference/libraries/?hl=zh-tw), Translation (https://cloud.google.com/translate/docs/reference/libraries/?hl=zh-tw), and Text-to-Speech (https://cloud.google.com/text-to-speech/docs/reference/libraries/?hl=zh-tw) client libraries.

To install the relevant client libraries, run the following commands from the terminal.

```
pip install --upgrade google-cloud-vision
pip install --upgrade google-cloud-translate
pip install --upgrade google-cloud-texttospeech
```

Setting up permissions for glossary creation

Creating Translation glossaries requires using a service account key with "Cloud Translation API Editor" (https://cloud.google.com/iam/docs/understanding-roles?hl=zh-tw#cloud-translation-roles) permissions.

To set up a service account key with "Cloud Translation API Editor" permissions:

1. On the Google Cloud **Service accounts** page (https://console.cloud.google.com/iam-admin/serviceaccounts/?hl=zh-tw), **Select a project**. Then, select **Create Service Account**. Designate the **Service account name** and **Create** the service account.
2. Under **Service account permissions**, click **Select a role**. Scroll to ***Cloud Translation*** and select ***Cloud Translation API Editor***. Select **Continue**.
3. Click **Create Key**, select **JSON**, and click **Create**.
4. From the `hybrid_glossaries` folder (#downloading_the_code_samples) in terminal, set the ***GOOGLE_APPLICATION_CREDENTIALS*** variable using the following command. Replace ***path_to_key*** with the path to the downloaded JSON file containing your new service account key.

LINUX OR MACOS | WINDOWS

```
export GOOGLE_APPLICATION_CREDENTIALS=path_to_key
```

Importing libraries

This tutorial uses the following system imports and client library imports.

translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py)

PLES/BLOB/MASTER/TRANSLATE/CLOUD-CLIENT/HYBRID_GLOSSARIES/HYBRID_TUTORIAL.PY)

```
import io
import os
import html

# Imports the Google Cloud client libraries
from google.api_core.exceptions import AlreadyExists
from google.cloud import translate_v3beta1 as translate
from google.cloud import vision
from google.cloud import texttospeech
```

Setting your project ID

You must associate a Google Cloud project (#prerequisites) with each request to a Google Cloud API. Designate your Google Cloud project (#prerequisites) by setting the `GCLOUD_PROJECT` environment variable from the terminal.

In the following command, replace ***project-id*** with your Google Cloud project ID. Run the following command from the terminal.

LINUX OR MACOS WINDOWS

```
export GCLOUD_PROJECT=project-id
```

This tutorial uses the following global project ID variable.

translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py)

PLES/BLOB/MASTER/TRANSLATE/CLOUD-CLIENT/HYBRID_GLOSSARIES/HYBRID_TUTORIAL.PY)

```
# extract GCP project id
PROJECT_ID = os.environ['GCLOUD_PROJECT']
```

Using Vision to detect text from an image

Use the Vision API to detect and extract text from an image. The Vision API uses Optical Character Recognition (OCR) (https://cloud.google.com/vision/docs/ocr/?hl=zh-tw) to support two text-detection features: detection of dense text, or `DOCUMENT_TEXT_DETECTION` (https://cloud.google.com/vision/docs/features-list?hl=zh-tw#text-detection), and sparse text detection, or `TEXT_DETECTION` (https://cloud.google.com/vision/docs/features-list?hl=zh-tw#document-text-detection-dense----------text--handwriting)

.

The following code shows how to use the Vision API `DOCUMENT_TEXT_DETECTION` (https://cloud.google.com/vision/docs/features-list?hl=zh-tw#text-detection) feature to detect text in a photo with dense text.

translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py)

PLES/BLOB/MASTER/TRANSLATE/CLOUD-CLIENT/HYBRID_GLOSSARIES/HYBRID_TUTORIAL.PY)

```
def pic_to_text(infile):
    """Detects text in an image file

    ARGS
    infile: path to image file

    RETURNS
    String of text detected in image
    """

    # Instantiates a client
    client = vision.ImageAnnotatorClient()

    # Opens the input image file
    with io.open(infile, 'rb') as image_file:
        content = image_file.read()

    image = vision.types.Image(content=content)

    # For dense text, use document_text_detection
    # For less dense text, use text_detection
    response = client.document_text_detection(image=image)
    text = response.full_text_annotation.text

    return text
```

Using Translation with glossaries

After extracting text from an image, use Translation glossaries (https://cloud.google.com/translate/docs/glossary?hl=zh-tw) to personalize the translation of the extracted text. Glossaries provide pre-defined translations that override the Cloud Translation API translations of designated terms.

Glossary use cases include:

- **Product names:** For example, 'Google Home' must translate to 'Google Home'.

- **Ambiguous words:** For example, the word 'bat' can mean a piece of sports equipment or an animal. If you know that you are translating words about sports, you might want to use a glossary to feed the Cloud Translation API the sports translation of 'bat', not the animal translation.
- **Borrowed words:** For example, 'bouillabaisse' in French translates to 'bouillabaisse' in English; the English language borrowed the word 'bouillabaisse' from the French language. An English speaker lacking French cultural context might not know that bouillabaisse is a French fish stew dish. Glossaries can override a translation so that 'bouillabaisse' in French translates to 'fish stew' in English.

Making a glossary file

The Cloud Translation API accepts TSV, CSV, or TMX glossary files. This tutorial uses a CSV file uploaded to Cloud Storage to define sets of equivalent terms.

To make a glossary CSV file:

1. **Designate the language of a column** using either iso-639-1 (https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes) or BCP-47 (https://tools.ietf.org/html/bcp47) language codes in the first row of the CSV file.

   ```
   fr,en,
   ```

2. **List pairs of equivalent terms** in each row of the CSV file. Separate terms with commas. The following example defines the English translation for several culinary French words.

   ```
   fr,en,
   chèvre,goat cheese,
   crème brulée,crème brulée,
   bouillabaisse,fish stew,
   steak frites,steak with french fries,
   ```

3. **Define variants of a word**. The Cloud Translation API is case-sensitive and sensitive to special characters such as accented words. Ensure that your glossary handles variations on a word by explicitly defining different spellings of the word.

```
fr,en,
chevre,goat cheese,
Chevre,Goat cheese,
chèvre,goat cheese,
Chèvre,Goat cheese,
crème brulée,crème brulée,
Crème brulée,Crème brulée,
Crème Brulée,Crème Brulée,
bouillabaisse,fish stew,
Bouillabaisse,Fish stew,
steak frites,steak with french fries,
Steak frites,Steak with french fries,
Steak Frites,Steak with French Fries,
```

4. Upload (https://cloud.google.com/storage/docs/uploading-objects/?hl=zh-tw) the glossary to a Cloud Storage bucket (https://cloud.google.com/storage/docs/creating-buckets/?hl=zh-tw). For the purposes of this tutorial, you do not need to upload a glossary file to a Cloud Storage bucket nor do you need to create a Cloud Storage bucket. Instead, use the publicly-available glossary file created for this tutorial to avoid incurring any Cloud Storage costs. Send the URI of a glossary file in Cloud Storage to the Cloud Translation API to create a glossary resource. The URI of the publicly-available glossary file for this tutorial is ***gs://cloud-samples-data/translation/bistro_glossary.csv*** (https://storage.cloud.google.com/cloud-samples-data/translation/bistro_glossary.csv?hl=zh-tw) . To download the glossary, click on the above URI link, but do not open it in a new tab.

Creating a glossary resource

In order to use a glossary, you must create a glossary resource with the Cloud Translation API. To create a glossary resource, send the URI of a glossary file in Cloud Storage to the Cloud Translation API.

Make sure that you are using a service account key with "Cloud Translation API Editor" (#setting_up_permissions_for_glossary_creation) permissions and make sure that you have set your project ID from the terminal (#setting_your_project_id).

The following function creates a glossary resource. With this glossary resource, you can personalize the translation request in the next step of this tutorial.

translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py)

PLES/BLOB/MASTER/TRANSLATE/CLOUD-CLIENT/HYBRID_GLOSSARIES/HYBRID_TUTORIAL.PY)

```
def create_glossary(languages, project_id, glossary_name, glossary_uri):
    """Creates a GCP glossary resource
    Assumes you've already manually uploaded a glossary to Cloud Storage

    ARGS
    languages: list of languages in the glossary
    project_id: GCP project id
    glossary_name: name you want to give this glossary resource
    glossary_uri: the uri of the glossary you uploaded to Cloud Storage

    RETURNS
    nothing
    """

    # Instantiates a client
    client = translate.TranslationServiceClient()

    # Designates the data center location that you want to use
    location = 'us-central1'

    # Set glossary resource name
    name = client.glossary_path(
        project_id,
        location,
        glossary_name)

    # Set language codes
    language_codes_set = translate.types.Glossary.LanguageCodesSet(
        language_codes=languages)

    gcs_source = translate.types.GcsSource(
        input_uri=glossary_uri)

    input_config = translate.types.GlossaryInputConfig(
        gcs_source=gcs_source)

    # Set glossary resource information
    glossary = translate.types.Glossary(
        name=name,
        language_codes_set=language_codes_set,
        input_config=input_config)
```

```
    parent = client.location_path(project_id, location)

    # Create glossary resource
    # Handle exception for case in which a glossary
    #  with glossary_name already exists
    try:
        operation = client.create_glossary(parent=parent, glossary=glossa
        operation.result(timeout=90)
        print('Created glossary ' + glossary_name + '.')
    except AlreadyExists:
        print('The glossary ' + glossary_name +
              ' already exists. No new glossary was created.')
```

Translating with glossaries

Once you create a glossary resource, you can use the glossary resource to personalize translations of text that you send to the Cloud Translation API.

The following function uses your previously-created glossary resource to personalize the translation of text.

translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py)

PLES/BLOB/MASTER/TRANSLATE/CLOUD-CLIENT/HYBRID_GLOSSARIES/HYBRID_TUTORIAL.PY)

```
def translate_text(text, source_language_code, target_language_code,
                   project_id, glossary_name):
    """Translates text to a given language using a glossary

    ARGS
    text: String of text to translate
    source_language_code: language of input text
    target_language_code: language of output text
    project_id: GCP project id
    glossary_name: name you gave your project's glossary
        resource when you created it

    RETURNS
    String of translated text
    """
```

```
# Instantiates a client
client = translate.TranslationServiceClient()

# Designates the data center location that you want to use
location = 'us-central1'

glossary = client.glossary_path(
    project_id,
    location,
    glossary_name)

glossary_config = translate.types.TranslateTextGlossaryConfig(
    glossary=glossary)

parent = client.location_path(project_id, location)

result = client.translate_text(
    parent=parent,
    contents=[text],
    mime_type='text/plain',  # mime types: text/plain, text/html
    source_language_code=source_language_code,
    target_language_code=target_language_code,
    glossary_config=glossary_config)

# Extract translated text from API response
return result.glossary_translations[0].translated_text
```

Using Text-to-Speech with Speech Synthesis Markup Language

Now that you have personalized a translation of image-detected text, you are ready to use the Text-to-Speech API. The Text-to-Speech API can create synthetic audio of your translated text.

The Text-to-Speech API generates synthetic audio from either a string of plain text or a string of text marked up with Speech Synthesis Markup Language (SSML) (https://cloud.google.com/text-to-speech/docs/ssml?hl=zh-tw). SSML is a markup language which supports annotating text with SSML tags (https://cloud.google.com/text-to-speech/docs/ssml?hl=zh-tw#support-for-ssml-elements). You can use SSML tags to influence how the Text-to-Speech API formats synthetic speech creation (https://cloud.google.com/text-to-speech/docs/ssml-tutorial?hl=zh-tw).

The following function converts a string of SSML to an MP3 file of synthetic speech.

translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py)

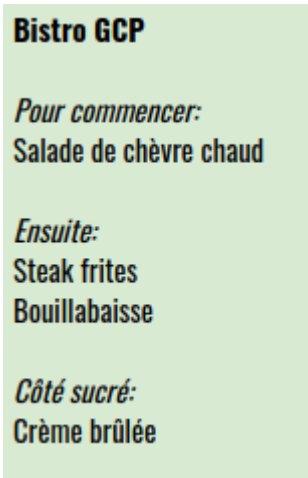
PLES/BLOB/MASTER/TRANSLATE/CLOUD-CLIENT/HYBRID_GLOSSARIES/HYBRID_TUTORIAL.PY)

```
def text_to_speech(text, outfile):
    """Converts plaintext to SSML and
    generates synthetic audio from SSML

    ARGS
    text: text to synthesize
    outfile: filename to use to store synthetic audio

    RETURNS
    nothing
    """

    # Replace special characters with HTML Ampersand Character Codes
    # These Codes prevent the API from confusing text with
    # SSML commands
    # For example, '<' --> '&lt;' and '&' --> '&amp;'
    escaped_lines = html.escape(text)

    # Convert plaintext to SSML in order to wait two seconds
    #   between each line in synthetic speech
    ssml = '<speak>{}</speak>'.format(
        escaped_lines.replace('\n', '\n<break time="2s"/>'))

    # Instantiates a client
    client = texttospeech.TextToSpeechClient()

    # Sets the text input to be synthesized
    synthesis_input = texttospeech.types.SynthesisInput(ssml=ssml)

    # Builds the voice request, selects the language code ("en-US") and
    # the SSML voice gender ("MALE")
    voice = texttospeech.types.VoiceSelectionParams(
        language_code='en-US',
        ssml_gender=texttospeech.enums.SsmlVoiceGender.MALE)

    # Selects the type of audio file to return
    audio_config = texttospeech.types.AudioConfig(
        audio_encoding=texttospeech.enums.AudioEncoding.MP3)
```

```
    # Performs the text-to-speech request on the text input with the sele
    # voice parameters and audio file type
    response = client.synthesize_speech(synthesis_input, voice, audio_con

    # Writes the synthetic audio to the output file.
    with open(outfile, 'wb') as out:
        out.write(response.audio_content)
        print('Audio content written to file ' + outfile)
```

Putting it all together

In the previous steps, you defined functions in **`hybrid_glossaries.py`** that use Vision, Translation, and Text-to-Speech. Now, you are ready to use these functions to generate synthetic speech of translated text from the following photo.



The following code calls functions defined in **`hybrid_glossaries.py`** to:

- create a Cloud Translation API glossary resource
- use the Vision API to detect text in the above image
- perform a Cloud Translation API glossary translation of the detected text
- generate Text-to-Speech synthetic speech of the translated text

translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/translate/cloud-client/hybrid_glossaries/hybrid_tutorial.py)

PLES/BLOB/MASTER/TRANSLATE/CLOUD-CLIENT/HYBRID_GLOSSARIES/HYBRID_TUTORIAL.PY)

```
def main():

    # Photo from which to extract text
    infile = 'resources/example.png'
    # Name of file that will hold synthetic speech
    outfile = 'resources/example.mp3'

    # Defines the languages in the glossary
    # This list must match the languages in the glossary
    #   Here, the glossary includes French and English
    glossary_langs = ['fr', 'en']
    # Name that will be assigned to your project's glossary resource
    glossary_name = 'bistro-glossary'
    # uri of .csv file uploaded to Cloud Storage
    glossary_uri = 'gs://cloud-samples-data/translation/bistro_glossary.c

    create_glossary(glossary_langs, PROJECT_ID,  glossary_name, glossary_

    # photo -> detected text
    text_to_translate = pic_to_text(infile)
    # detected text -> translated text
    text_to_speak = translate_text(text_to_translate, 'fr', 'en',
                                   PROJECT_ID, glossary_name)
    # translated text -> synthetic audio
    text_to_speech(text_to_speak, outfile)


if __name__ == '__main__':
    main()
```

Running the code

To run the code, enter the following command in terminal in your cloned hybrid_glossaries directory (#downloading_the_code_samples):

```
python hybrid_tutorial.py
```

The following output appears:

```
Created glossary bistro-glossary.
Audio content written to file resources/example.mp3
```

After running **`hybrid_glossaries.py`**, navigate into the ***resources*** directory from the ***hybrid_glossaries*** directory. Check the resources directory for an **`example.mp3`** file.

Listen to the following audio clip to check that your **`example.mp3`** file sounds the same.

0:00 / 0:23

Troubleshooting error messages

- `403 Cloud IAM permission 'cloudtranslate.glossaries.create' denied.`

  Using a service account key without "Cloud Translation API Editor" permissions (#setting_up_permissions_for_glossary_creation) raises this exception.

- `KeyError: 'GCLOUD_PROJECT'`

  Not setting your GCLOUD_PROJECT variable (#setting_your_project_id) generates this error.

- `400 Invalid resource name project id`

  Either using a glossary name which contains characters other than lowercase letters, digits, periods, a colon, or hyphens, or using a service account key without "Cloud Translation API Editor" permissions (#setting_up_permissions_for_glossary_creation) raises this exception.

- `File ***filename*** was not found.`

  Setting the GOOGLE_APPLICATION_CREDENTIALS variable (#setting_up_permissions_for_glossary_creation) to an invalid filepath raises this exception.

- `Could not automatically determine credentials. Please set GOOGLE_APPL`

  Not setting the GOOGLE_APPLICATION_CREDENTIALS variable (#setting_up_permissions_for_glossary_creation) raises this exception.

- `Forbidden: 403 POST API has not been used or is disabled`

  Calling the Cloud Translation API, the Cloud Vision API, or the Text-to-Speech API without enabling their APIs (https://cloud.google.com/apis/docs/getting-started?hl=zh-tw#enabling_apis) generates this warning.

- `AttributeError: 'module' object has no attribute 'escape'`

  Python 2.7.10 or earlier is not compatible with `HTML`. To fix this error, use a Python virtual environment (https://docs.python.org/3/library/venv.html). The virtual environment will use the newest version of Python.

- `UnicodeEncodeError`

  Python 2.7.10 or earlier is not compatible with `HTML`. To fix this error, use a Python virtual environment (https://docs.python.org/3/library/venv.html). The virtual environment will use the newest version of Python.

Cleaning up

Use the Google Cloud Console (https://console.cloud.google.com/?hl=zh-tw) to delete your project if you do not need it. Deleting your project prevents incurring additional charges to your Google Cloud Platform account for the resources used in this tutorial.

Deleting your project

1. In the Cloud Console (https://console.cloud.google.com/?hl=zh-tw), go to the Projects page.
2. In the project list, select the project you want to delete and click **Delete**.
3. In the dialog box, type the project ID, and click **Shut down** to delete the project.

What's next

Congratulations! You just used Vision OCR to detect text in an image. Then, you created a Translation glossary and performed a translated with that glossary.

Afterwards, you used Text-to-Speech to generate synthetic audio of the translated text.

To build on your knowledge of Vision, Translation, and Text-to-Speech:

- Make your own glossary. Learn how to create a Cloud Storage bucket (https://cloud.google.com/storage/docs/creating-buckets/?hl=zh-tw) and to upload your glossary CSV file to the bucket (https://cloud.google.com/storage/docs/uploading-objects/?hl=zh-tw).
- Experiment with other ways to use Translation glossaries (https://cloud.google.com/translate/docs/glossary?hl=zh-tw).
- Learn how to use Cloud Storage with Cloud Vision OCR (https://cloud.google.com/functions/docs/tutorials/ocr/?hl=zh-tw).
- Learn more about how to use SSML with Text-to-Speech (https://cloud.google.com/text-to-speech/docs/ssml-tutorial/?hl=zh-tw).
- Learn how to use the Vision API `imageContext` field (https://cloud.google.com/vision/docs/handwriting?hl=zh-tw#specify_the_language_optional) to pass along additional context about a photo when using Vision OCR.
- Explore community tutorials (https://cloud.google.com/community/tutorials/?hl=zh-tw).

上次更新：十二月 4, 2019。