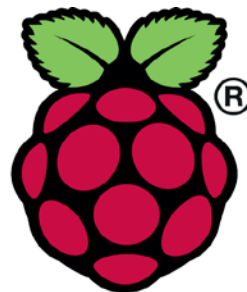
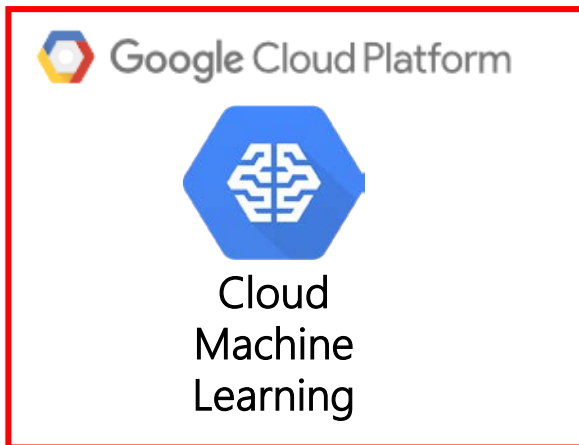


抜粋版

はじめてのクラウドAI開発キット ～グーグルのAIノウハウを最大活用、AI技術者不足に対応～ 開発編



Raspberry Pi

スペクトラム・テクノロジー株式会社

<https://spectrum-tech.co.jp>

sales@spectrum-tech.co.jp

開発編 目次

	ページ	
• Raspberry Pi運用マニュアル		
1. RaspberryPiについて	4	
2. Linux基本コマンド	5	
3. RaspberryPi基本操作	6	
4. 日常運用(ウイルススキャン、更新)	7	
• クラウドAI公式ドキュメント	9	
• GCP設定(Google cloud platform)		
1. アカウント登録	10	抜粋版のため、 ページは正しく ありません
2. プロジェクト登録	12	
3. API設定	13	
4. APIキーの発行	15	
5. サービスアカウントキーの発行	16	
6. IAMと管理	18	
• クラウドAI開発キット		
1. 開発キット全体像	19	
2. AI活用例	20	
3. 開発キットプログラム一覧	22	
4. 使用方法	26	

開発編 目次

• クラウドAI開発キット

4. 使用方法

	ページ
① Cloud vision:画像認識	
• 1_OCR(文字認識)	26
• API利用状況確認	33
• 2_face(顔検出)	34
• 3_label(ラベル検出)	38
• 4_web(web検出)	41
• 5_safe(不適切画像検出)	44
• 6_logo(ロゴ検出)	47
• 7_landmark(ランドマーク検出)	50
• 8_prop(プロパティ検出)	53
• 9_crop(切り出し検出)	56
• 複数画像同時処理	58
② Cloud speech:音声認識	
• transcribe(短時間音声認識)	62
• Google cloud storageへのアップロード	63
• transcribe_async(非同期音声認識)	66
• transcribe_stream_mic(マイク音声認識)	67
③ Cloud Natural language:自然言語解析	
• sentiment(感情分析)	70
• syntax(構文解析)	72
• entity(エンティティ分析)	73
• classify(カテゴリ分類):英語のみ	74
④ Cloud translate:翻訳	75
⑤ Cloud video:動画認識	77

抜粋版のため、ページ
は正しくありません

RaspberryPi運用マニュアル

1. Raspberry Piについて

既に全世界で1000万台以上販売された手のひらサイズのコンピュータです。
LinuxベースのRasbianOSで動作しております。

2. Linux基本コマンド

① システム関係

- 起動: 電源を入れると自動で起動します。
- 再起動: # reboot
又は、menu>shutdown>reboot; 左上のメニューから
- 終了: # shutdown
又は、menu>shutdown>shutdown; 左上のメニューから
- ログアウト # exit
又は、menu>shutdown>logout; 左上のメニューから
- **日本語／英語の入力切替**: キーボードのCTLとjを同時に押します(コントロール: 左下とj)

RaspberryPi運用マニュアル

2. Linux基本コマンド

② ディレクトリ操作、コピー、移動、削除

root@raspberrypi:~# cd /home/pi/Documents	ディレクトリの切り替え
root@raspberrypi:/home/pi/Documents# ls	ファイルとディレクトリの表示(表示したら操作したいファイルを右クリックでコピーして操作します)
root@raspberrypi:~# cp ファイル名 ディレクトリ	配下のディレクトリのファイルを別のディレクトリへコピー
root@raspberrypi:~# mv ファイル名 ディレクトリ	配下のディレクトリのファイルを別のディレクトリへ移動
root@raspberrypi:~# rm ファイル名	ファイルの削除
root@raspberrypi:~# rm -help	コマンドのオプションが分からない場合は、ヘルプで問い合わせる。すべてのコマンド共通(マイナスを2個とhelp)

③ ユーザ権限、プロセス他

pi@raspberrypi:~ \$ su -	スーパーユーザ(root)に切り替え、パスワードを入力
root@raspberrypi:~# ps a	現状の動いているプロセスを表示
root@raspberrypi:~# kill	特定のプロセスを強制終了
root@raspberrypi:~# apt-get install pkg	パッケージのインストールなどに使用
root@raspberrypi:~# date	日付、時間の設定を行います。
root@raspberrypi:~# leafpad /etc/network/interfaces	インタフェースに記述している内容を変更します。Viよりも使いやすいです。

④ モジュール、usb、メモリ、HDDなどの表示

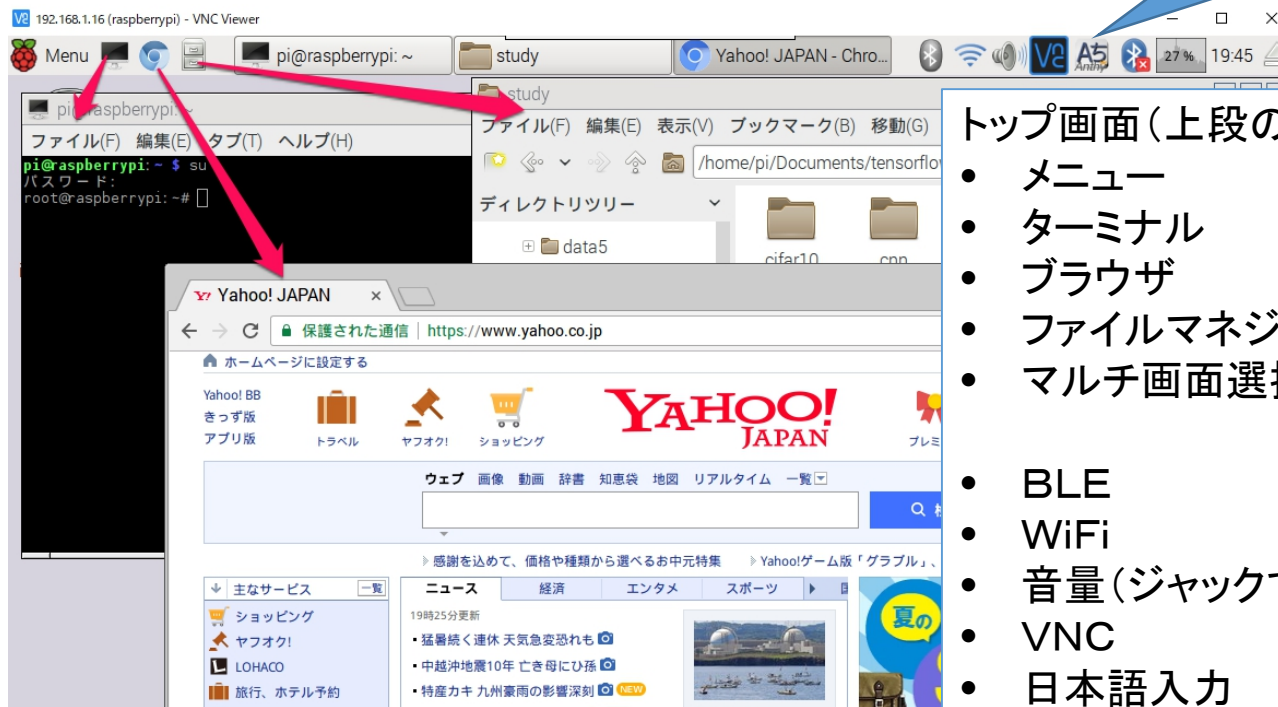
root@raspberrypi:~# lsmod	linuxのモジュールリスト表示
root@raspberrypi:~# lsusb	usbのデバイス表示
root@raspberrypi:~# free -mt	メモリ使用状態表示
root@raspberrypi:~# df	HDD(マイクロSD)の使用状態表示

RaspberryPi運用マニュアル

3. Raspberry Piの基本操作

① 表示画面と内容

Anthyが出ない場合は、一度 Japaneseを選択後、再度 Anthyを選択してください。



トップ画面(上段のタスクバーで選択)

- メニュー
- ターミナル
- ブラウザ
- ファイルマネージャ
- マルチ画面選択

- BLE
- WiFi
- 音量(ジャックで聴けます)
- VNC
- 日本語入力
- BLE
- 回線効率
- 時刻

クラウドAI公式ドキュメント

以下のドキュメントで学習しながら、プログラムを実行してください。より理解が深まります。

- Cloud vision: 画像認識

<https://cloud.google.com/vision/docs/?authuser=1&hl=ja>

- Cloud speech: 音声認識

<https://cloud.google.com/speech/docs/?authuser=1&hl=ja>

- Cloud Natural language: 自然言語解析

<https://cloud.google.com/natural-language/docs/?authuser=1&hl=ja>

- Cloud translate: 自動翻訳

<https://cloud.google.com/translate/docs/?authuser=1&hl=ja>

- Cloud video: 動画認識

<https://cloud.google.com/video-intelligence/docs/?authuser=1&hl=ja>

GCP設定 (Google cloud platform)

1. アカウント登録

- ・GCPにアクセスして、無料トライアルを申しこみ、アカウント登録を実施します

<https://console.cloud.google.com/>

無料トライアル終了後は、有料になります。各社で料金をチェックしてください。

The screenshot shows the Google Cloud Platform website. On the left, a blue button labeled '無料トライアル' (Free Trial) is highlighted with a red box. An arrow points from this button to a confirmation dialog box on the right. The dialog box is titled 'Google Cloud Platform の無料トライアル' and contains the following text:

国
日本

受諾
新機能のお知らせ、パフォーマンスに関するアドバイス、フィードバック調査、特典に関する最新情報をメールで受け取ります。
 はい いいえ

Google Cloud Platform 無料トライアルの利用規約を讀んだうえで内容に同意します。
続行するには「はい」を選択する必要があります。
 はい いいえ

同意して続行

On the right side of the dialog box, there is a list of terms and conditions:

- すべての Cloud Platform サービスへのアクセス
アプリや、ウェブサイト、サービスの構築と実行に必要な Firebase や Google Maps API などすべて使用できます。
- \$300 相当のクレジットを無料でご提供
ご登録いただく、Google Cloud Platform で今後 12 か月間ご利用いただける \$300 のクレジットを獲得できます。
- 無料トライアル期間が終了しても、自動的に請求されることはありません
ロボットによる登録ではないことを確認するため、クレジットカード番号を確認しております。有料アカウントに手動でアップグレードしない限り、課金されることはありません。

GCP設定 (Google cloud platform)

1. アカウント登録

- ・住所、会社名、氏名、電話、クレジットカード情報を入力して、申し込みます

Google Cloud Platform

Google Cloud Platform の無料トライアル Google

お客様情報

アカウントの種類 ⓘ

ビジネス

名前と住所 ⓘ

郵便番号 ⓘ

郵便番号を入力してください

都道府県

市区郡

住所 1 行目

住所 2 行目

企業/組織名

企業/組織名

名前

メインの連絡先 ⓘ

お支払いタイプ

毎月の自動支払い

このサービスの費用を毎月 1 回定期的にお支払いいただけます。お支払い期限になると自動的に請求が行われます。

お支払い方法 ⓘ

カードの詳細

クレジット (デビット) カードの住所は上記と同じ

無料トライアルを開始

すべての Cloud Platform サービスへのアクセス

アプリや、ウェブサイト、サービスの構築と実行に必要な Firebase や Google Maps API などがすべて使用できます。

\$300 相当のクレジットを無料でご提供

ご登録いただくと、Google Cloud Platform で今後 12 か月間ご利用いただける \$300 のクレジットを獲得できます。

無料トライアル期間が終了しても、自動的に請求されることはありません

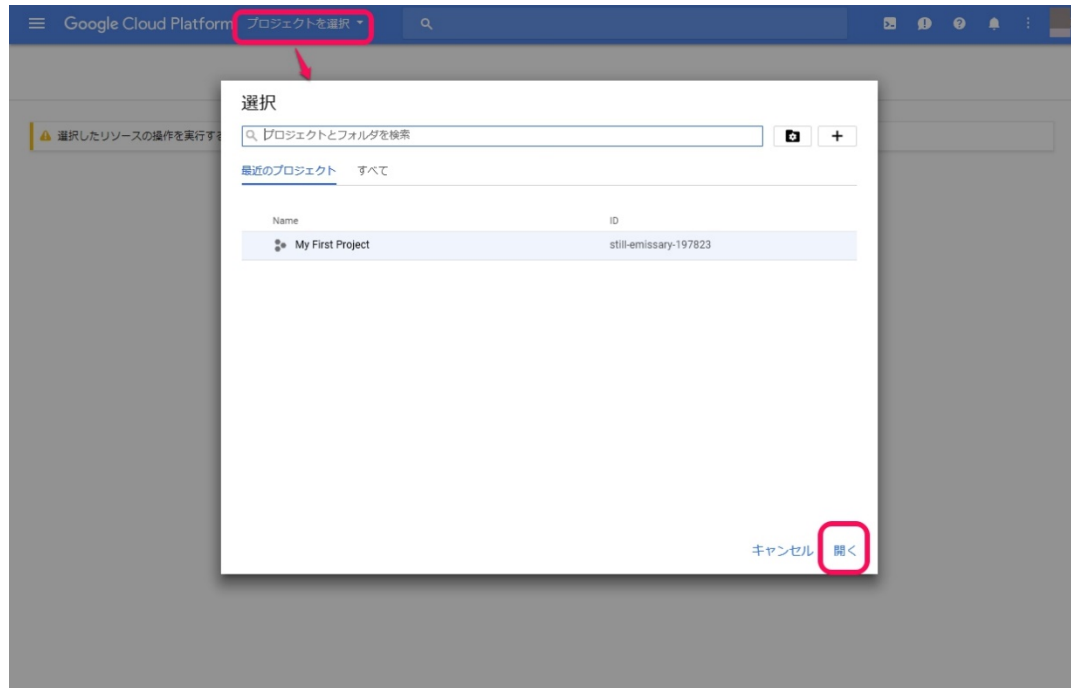
ロボットによる登録ではないことを確認するため、クレジットカード番号を確認しております。有料アカウントに手動でアップグレードしない限り、課金されることはありません。

プライバシーポリシー | よくある質問

GCP設定 (Google cloud platform)

2.プロジェクト登録

- ・GCPのホームからプロジェクトを選択して、プロジェクト名を記入して、登録します。
- ・プロジェクト名とプロジェクトIDは作成後は、修正できません。



GCP設定 (Google cloud platform)

3.API設定

- ・使用するAPIを有効にします。
- ・APIとサービス>ダッシュボード>APIとサービスの有効化

The screenshot shows the Google Cloud Platform console interface. In the left-hand navigation menu, the 'API とサービス' (API and Services) option is highlighted with a red box. A blue arrow points from this menu item to the 'ダッシュボード' (Dashboard) link in the top navigation bar, which is also highlighted with a red box. The main content area displays the '有効化された API とサービス' (Enabled APIs and Services) dashboard, which includes a table of active APIs and their status.

API	リクエスト数	エラー	エラー率	レイテンシ, 中央値	レイテンシ, 98%
BigQuery API	-	-	-	-	- 無効にする
Google Cloud APIs	-	-	-	-	- 無効にする
Google Cloud Datastore API	-	-	-	-	- 無効にする
Google Cloud SQL	-	-	-	-	- 無効にする
Google Cloud Storage	-	-	-	-	- 無効にする
Google Cloud Storage JSON API	-	-	-	-	- 無効にする
Google Service Management API	-	-	-	-	- 無効にする
Stackdriver Debugger API	-	-	-	-	- 無効にする

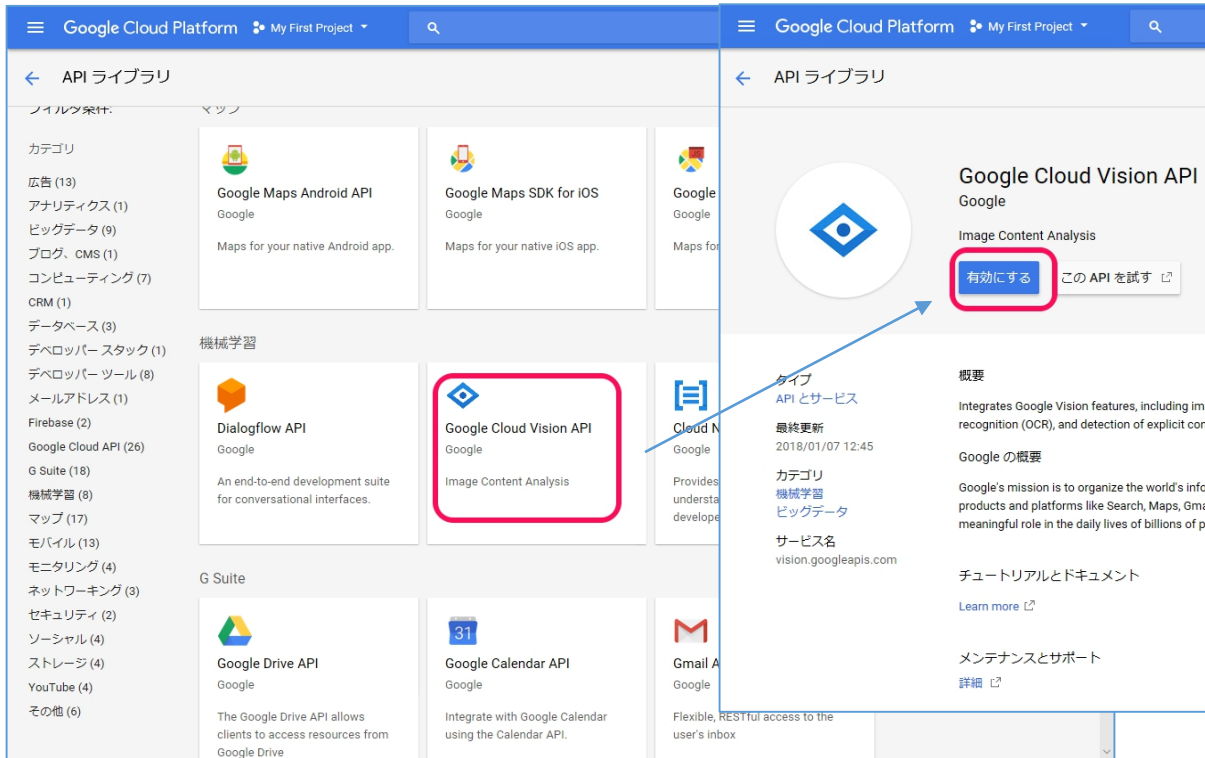
GCP設定 (Google cloud platform)

3.API設定

- APIライブラリから機械学習 > Cloud vision APIを選択。
- Cloud vision APIの有効にします。

有効にするAPI

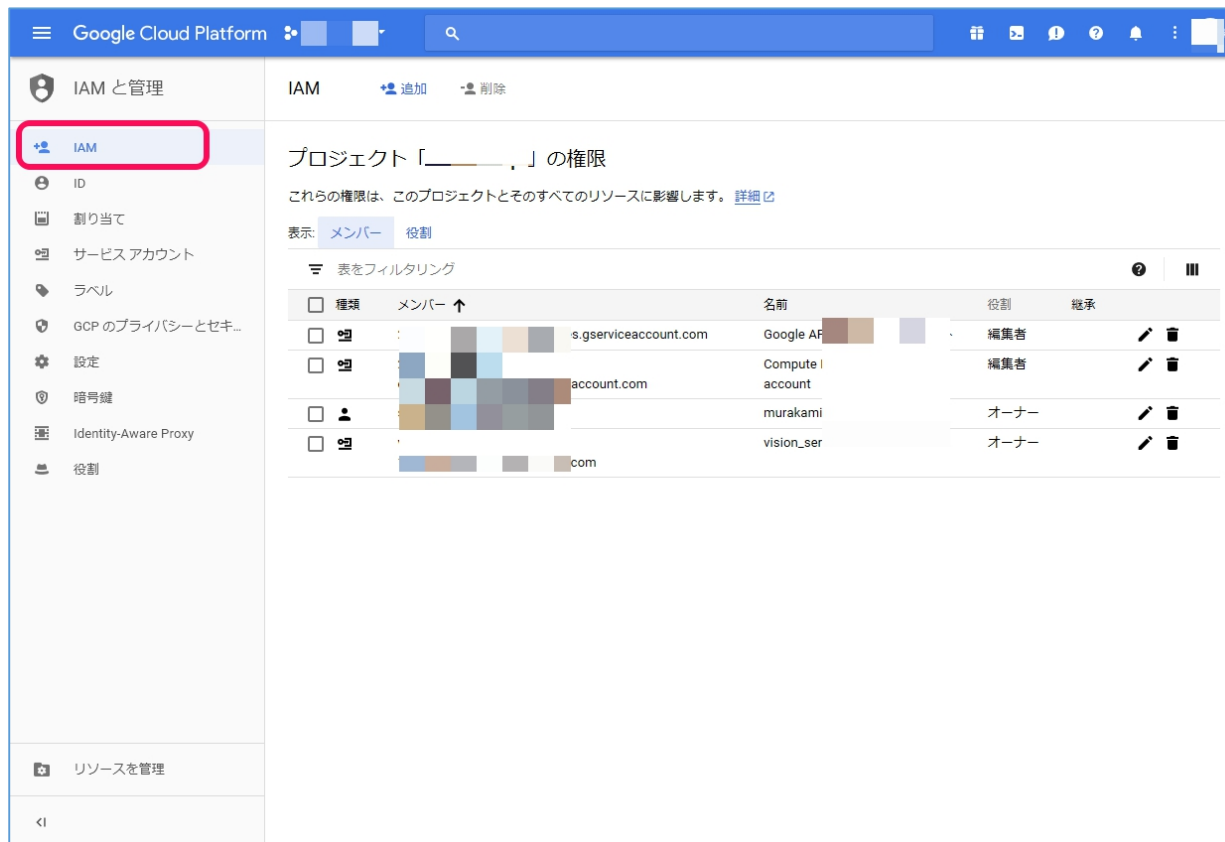
- Cloud vision API
- Cloud speech API
- Cloud natural language API
- Cloud translate API
- Cloud Video API(β)



GCP設定 (Google cloud platform)

6.IAMと管理

- ・GCPにアクセスするアカウントなどを一元的に管理するサービスです。
- ・プロジェクトのオーナーの方は、不要なアカウントがないか定期的の確認しましょう。

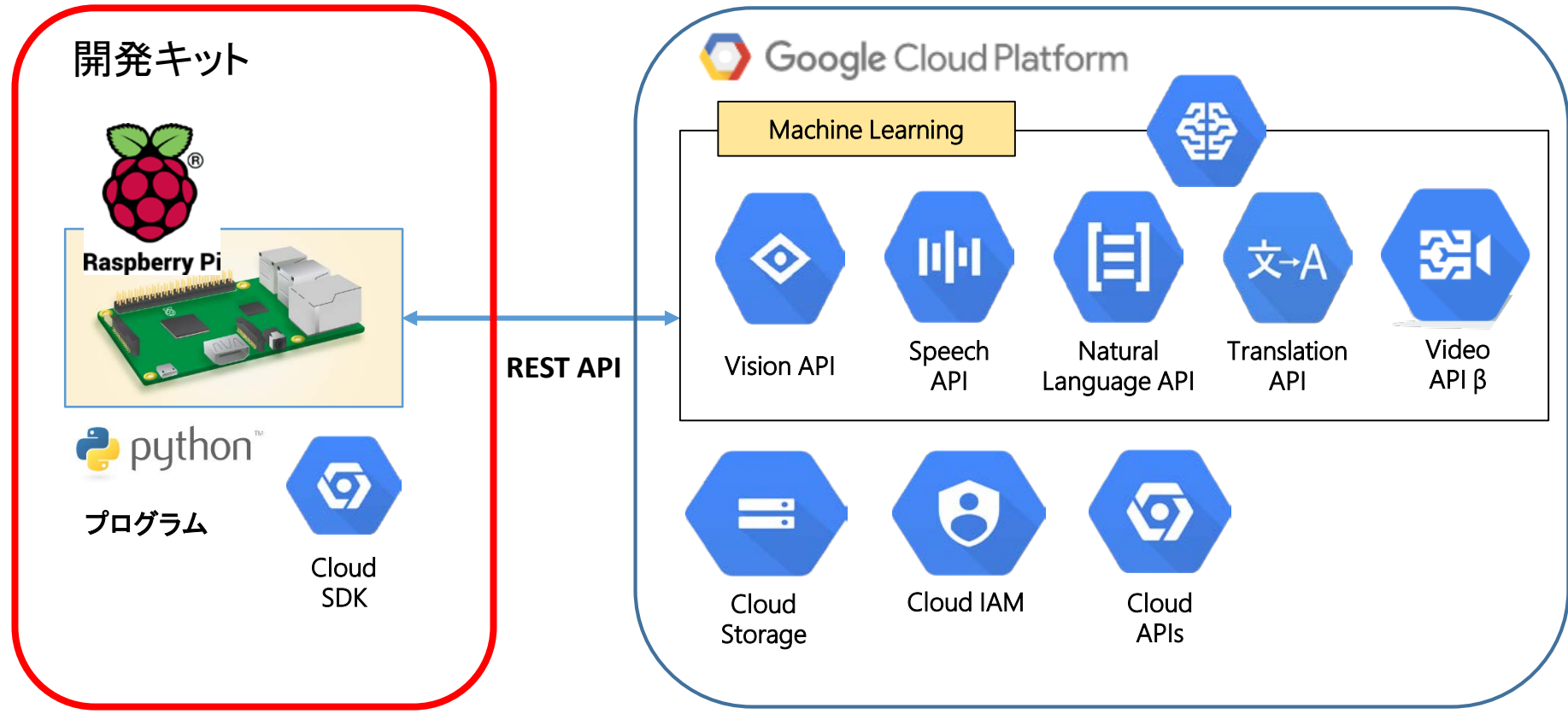


The screenshot shows the Google Cloud Platform IAM management interface. The left sidebar is titled 'IAM と管理' and includes options like 'IAM', 'ID', '割り当て', 'サービスアカウント', 'ラベル', 'GCP のプライバシーとセキュリティ', '設定', '暗号鍵', 'Identity-Aware Proxy', and '役割'. The 'IAM' option is highlighted with a red box. The main content area is titled 'IAM' and shows the permissions for a project. It includes a table of members and their roles.

種類	メンバー ↑	名前	役割	継承
<input type="checkbox"/>	Google AF	s.serviceaccount.com	編集者	
<input type="checkbox"/>	Compute I account	account.com	編集者	
<input type="checkbox"/>	murakami		オーナー	
<input type="checkbox"/>	vision_ser		オーナー	

クラウドAI開発キット

1. 開発キット全体像



ST Spectrum Technology

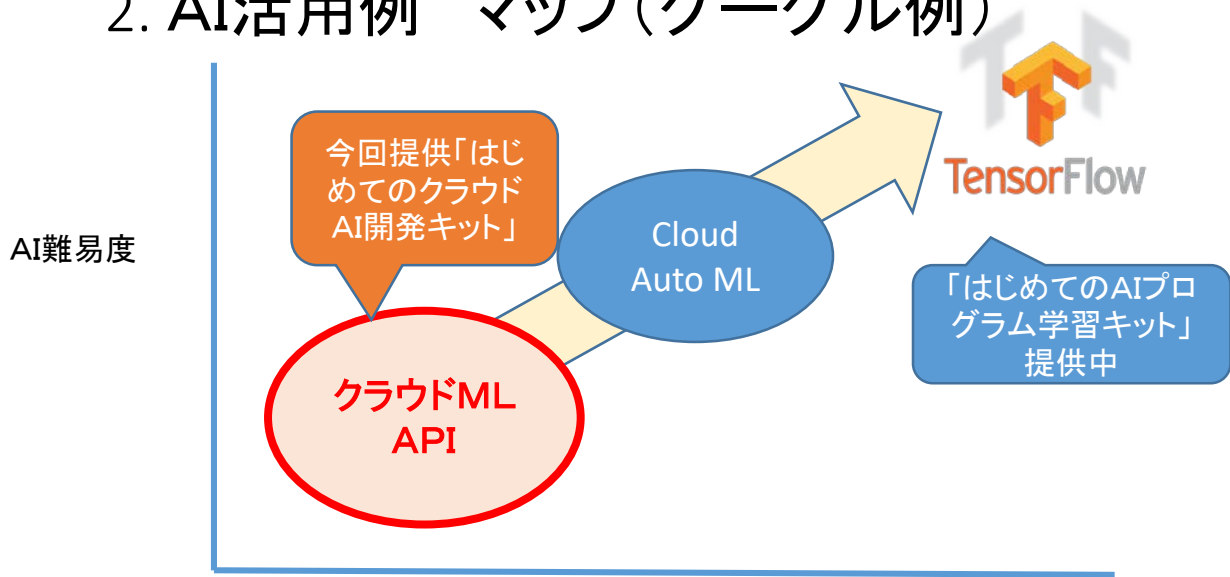
クラウドAI開発キット

2. AI活用例

カテゴリ	活用例	要素技術	
画像認識	<ul style="list-style-type: none"> ・OCRでの読み取り代替 ・写真のテキスト検出 ・顔検出による人数カウント ・不適切画像検出 ・類似デザイン検出 ・画像からWebサイト検出 ・画像へのインデックス付け 	Ocr Ocr Face Safe Web label	
音声認識	<ul style="list-style-type: none"> ・自動議事録作成 ・動画、音声ファイルからの文字起こし 	Transcribe_stream Transcribe_async	
自然言語解析	<ul style="list-style-type: none"> ・感情分析(音声認識との組合せ) ・エンティティ認識 	Sentiment+Transcribe_async entity	
自動翻訳	<ul style="list-style-type: none"> ・多言語翻訳 ・音声認識し、多言語で文字出力 	Snippet Translate+transcribe_async	
動画認識	<ul style="list-style-type: none"> ・不適切動画検出 ・ラベル検出 ・音声文字起こし 	Adult Label Transcribe_async	

クラウドAI開発キット

2. AI活用例 マップ(グーグル例)



一般向け

使用スキル

専門家向け

サービス名	特徴	考慮事項
クラウドML API	<ul style="list-style-type: none"> • グーグルのノウハウ(学習済モデル)を利用 • プログラム作成が簡単 • AI技術者が不要 	<ul style="list-style-type: none"> • カスタマイズが難しい
Cloud Auto ML	<ul style="list-style-type: none"> • クラウドAIとTensorflowの中間でカスタマイズが可能 • 高度なAI技術者が不要 	
Tensorflow	<ul style="list-style-type: none"> • 自由にモデルを作成可能 	<ul style="list-style-type: none"> • 学習データはユーザが準備 • AI技術者が必要

クラウドAI開発キット

3. 開発キットプログラム一覧

A: 実用可能
B: チャレンジ
C: 試験段階

クラウドAI名	プログラム名	機能内容	実用度 (当社評価)	信頼度 (当社評価)	備考
Cloud vision api (画像認識)	OCR(文字認識)	画像内のテキストを検出、抽出できます。幅広い言語がサポートされており、言語の種類も自動で判別されます。	A	90%	手書き文字はC:10%位
	Face(顔検出)	画像に含まれる複数の人物の顔を検出できます。感情や帽子の着用といった主要な顔の属性についても識別されます。	A	90%	特定の人顔認識はできません
	Label(ラベル検出)	乗り物や動物など、画像に写っているさまざまなカテゴリの物体を検出できます。	A	90%	
	Web(ウェブ検出)	類似の画像をインターネットで検索できません	A	90%	
	Safe(不適切画像検出)	アダルトコンテンツや暴力的コンテンツなど、画像に含まれる不適切なコンテンツを検出できます	A	90%	
	Logo(ロゴ検出)	画像に含まれる一般的な商品ロゴを検出できます	C	10%	文字のないロゴは検出できない
	Landmark(ランドマーク検出)	画像に含まれる一般的な自然のランドマークや人工建造物を検出できます。	B	50%(画像次第)	
	Prop(画像属性検出)	画像のドミナントカラーや切り抜きのヒントなど、画像の一般的な属性を検出できます	B		
	Crop(画像切り抜き)	認識できる画像に切り抜きできます。画像の最小化	B		
	multipage	複数画像を処理できるプログラム			
	loadjson	Json形式のデータを読み出すプログラム			

http方式とライブラリ方式の二つ準備

クラウドAI開発キット

3. 開発キットプログラム一覧

A: 実用可能
B: チャレンジ
C: 試験段階

クラウドAI名	部品名	機能内容	実用度(主観)	信頼度(主観)	備考
Cloud speech api (音声認識、多言語対応)	Transcribe(同期音声認識)	ローカルに保存されている、短い音声(1分未満)を処理し、認識されたテキストを出力します。	A	70%	固有名詞が難しい
	Transcribe_async(非同期音声認識)	Google Cloud Storage に保存されている、1分より長い音声を認識し、テキストで出力します。 動画から音声テキストの検出も可能	A	70%	
	Transcribe_stream(ストリーミング音声認識)	ローカルのリアルタイムでストリーミング音声認識します			ローカルなので上記の部品で代用可能
	Transcribe_stream_mic(マイク入力ストリーミング音声認識)	マイク入力で、リアルタイムで音声認識を行い、テキストを出力します。びっくりする能力。是非お試しください。 Googleドキュメントの音声入力と同じです。(*1)	A	70%	すごいです。
	pyaudio	マイクの試験用プログラム			

(*1)USBマイクは付いておりませんので、お客様で準備してください

クラウドAI開発キット

3. 開発キットプログラム一覧

A: 実用可能
 B: チャレンジ
 C: 試験段階

クラウドAI名	部品名	機能内容	実用度(主観)	信頼度(主観)	備考
Cloud Natural language api(自然言語処理、多言語)	Sentiment(感情分析)	テキストのブロック内で示されている全体的な感情を読み取ることができます。	B		
	Syntax(構文解析)	トークンと文の抽出、品詞(PoS)の特定、各文の係り受け解析ツリーの作成が可能です。	B		文字ずれ発生中:原因不明
	Entity(エンティティ認識)	エンティティとラベル(人、組織、場所、イベント、商品、メディアなど)を特定できます	A	90%	
	Classify(カテゴリ分類)英語のみ	事前定義された700以上のカテゴリでドキュメントを分類できます	A	90%	英語のみ
Cloud translate api(翻訳、多言語)	Quickstart(テキスト翻訳)	グーグルの自動翻訳。多言語対応。プログラム上で言語を指定	A	70%	
	Snippet(テキスト翻訳、多言語)	グーグルの自動翻訳。多言語を翻訳時に指定。	A	70%	

クラウドAI開発キット

3. 開発キットプログラム一覧

A: 実用可能
 B: チャレンジ
 C: 試験段階

クラウドAI名	部品名	機能内容	実用度(主観)	信頼度(主観)	備考
Cloud video api beta (動画認識)	Label(ラベル認識)	「犬」、「花」、「車」などの動画内のエンティティを検出します	A	70%	
	adult(不適切動画検出)	アダルトコンテンツや暴力的コンテンツなど、画像に含まれる不適切なコンテンツを検出できます	A	70%	
	analyze(統合型分析)	Shot: 動画内のシーンの変更を検出します Adult: 不適切シーンの検出	B		
	Face(顔認識)	プログラムはありますが、エラー。プライバシー保護のため	C		エラー
	音声認識	Speech>transcribe_asyncにて、動画の音声のみ取り出して、文字起こしが可能	A	70%	便利

クラウドAI開発キット

4.使用方法

① Cloud vision>1_OCR(文字認識)

- テキスト認識: ocr1.py (画像からテキスト抽出)
 - APIキーの設定
 - Windows PC>ネットワーク>RaspberryPi>Pi>Documents>cloud vision>1_ocrをクリック
 - Ocr1.pyをwindows PCのローカルにコピーして、**さくらエディタ**などでocr1.pyを開きます。
 - 17行のstr_api_key="xxxxx"にGCP設定>4.APIキー設定で取得したキー(15ページ)を貼り付けます
 - また、画像ファイル名は、7行目です。適宜書き換えて使用してください。
 - また、データ出力は、53行目になります。デフォルトは、data1.jsonです。
 - 修正が完了したら、再度raspberry Piにocr1.pyを上書きしてください。

```
1 #!/usr/bin/python
2 #coding:utf-8
3 import base64
4 import json
5 from requests import Request, Session
6
7 path = "choushi3.png"
8
9 def recognize_captcha(str_image_path):
10     bin_captcha = open(str_image_path, 'rb').read()
11
12     str_encode_file = base64.b64encode(bin_captcha)
13     #str_encode_file = base64.b64encode(bin_captcha).decode("utf-8")
14
15     str_url = "https://vision.googleapis.com/v1/images/annotate?key="
16
17     str_api_key = "XXXXXXXXXXXXXXXXXXXX"
18
19     str_headers = {'Content-Type': 'application/json'}
20
21     str_json_data = {
22         'requests': [
23             {
24                 'image': {
25                     'content': str_encode_file
26                 },
27                 'features': [
28                     {
29                         'type': "TEXT_DETECTION",
30                         'maxResults': 10
31                     }
32                 ]
33             }
34         ]
35     }
36
37     print("begin request")
38     obj_session = Session()
39     obj_request = Request("POST",
40                           str url + str api key,
```

クラウドAI開発キット

4.使用方法

① Cloud vision>1_OCR(文字認識)

- テキスト認識: ocr1.py(画像からテキスト抽出)
 - Raspberry piからocr1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata1.jsonの双方にでます。

コマンド

```
# cd /home/pi/Documents/gcp_api/cloud_vision/1_ocr
# python ocr1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>

The screenshot shows a Raspberry Pi terminal window with the following command and output:

```
pi@raspberrypi: ~
pi@raspberrypi: ~ $ su -
root@raspberrypi: ~# cd /home/pi/Documents/gcp_api/cloud_vision/1_ocr
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# ls
choushi3.png data2.json handwriting1.jpg ocr1.py ocr3.py receipt1.jpg
data1.json data3.json loadjson.py ocr2.py ocr4.py
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# python ocr1.py
begin request
end request
{
  "responses": [
    {
      "textAnnotations": [
        {
          "locale": "en",
          "description": "1625\n",
          "boundingPoly": {
            "vertices": [
              {
                "x": 128,
                "y": 222
              },
              {
                "x": 170,
```

The Google Cloud Vision interface shows the 'Text' tab selected, displaying the detected text '1625' from an image of a runner. The interface also includes a 'Learn More About Natural Language API' button.

クラウドAI開発キット

4.使用方法

① Cloud vision>1_OCR(文字認識)

- テキスト認識: ocr1.py(画像からテキスト抽出)
 - jsonデータを、loadjson.pyでテキストのみ取り出します。
 - 読み出すデータは5行目に定義。
 - 必要に応じて書き換えてください。

コマンド

```
# cd /home/pi/Documents/gcp_api/cloud_vision/1_ocr
# python loadjson.py
```

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
}
  }
  "blockType": "TEXT"
}
  }
  }
  }
  "text": "1625\n"
}
}
1625
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# ^C
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# ls
choushi3.png data2.json handwriting1.jpg ocr1.py ocr3.py receipt1.jpg
data1.json data3.json loadjson.py ocr2.py ocr4.py
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# python loadjson.py
1625
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr#
```

テキスト出力結果

```
デスクトップ#murakami_leno#ビジネス#技術関係#deep_learning#gcp_ai#cloud_vision#1_ocr#loadjson.py - sakura 2.2.0.1
ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)
1 #!/usr/bin/python
2 #coding:utf-8
3 import json
4
5 data = open('data1.json', 'r')
6 json_str = json.load(data) #json file load
7 str_dic = {} #str to dictionary
8
9 for i in range(5):
10     str_dic["text"] = json_str[i]["text"]
```

クラウドAI開発キット

4.使用方法

① Cloud vision>1_OCR(文字認識)

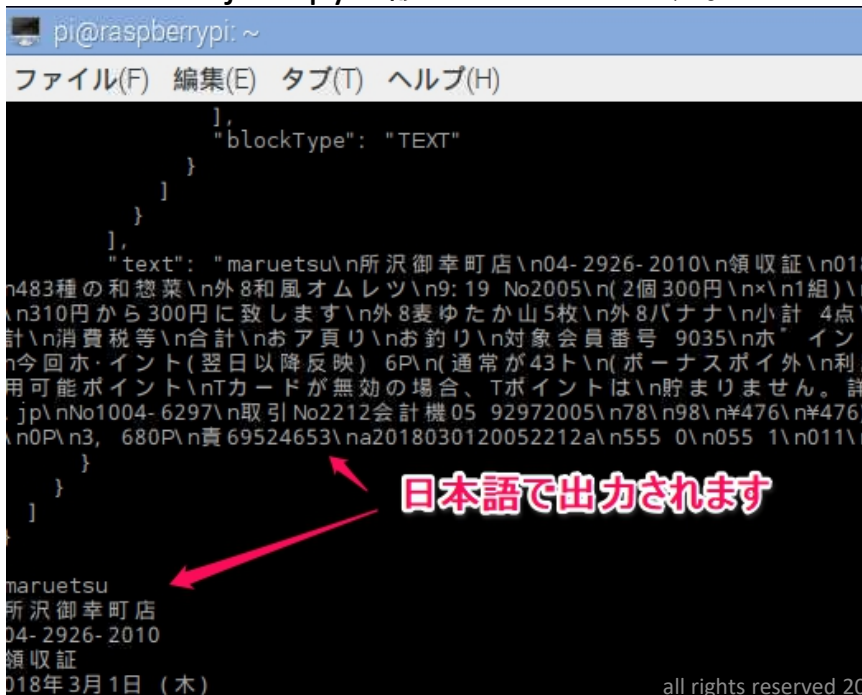
- テキスト認識: ocr2.py(文字読み取り)
 - 同様にApi キーを設定します。
 - Raspberry piからocr2.pyを実行します。
 - 出力は、piの画面上とファイルのdata2.jsonの双方にです。
 - loadjson.pyで読み出しできます。

コマンド

```
# cd /home/pi/Documents/gcp_api/cloud_vision/1_ocr
# python ocr2.py
# python loadjson.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>



クラウドAI開発キット

4.使用方法

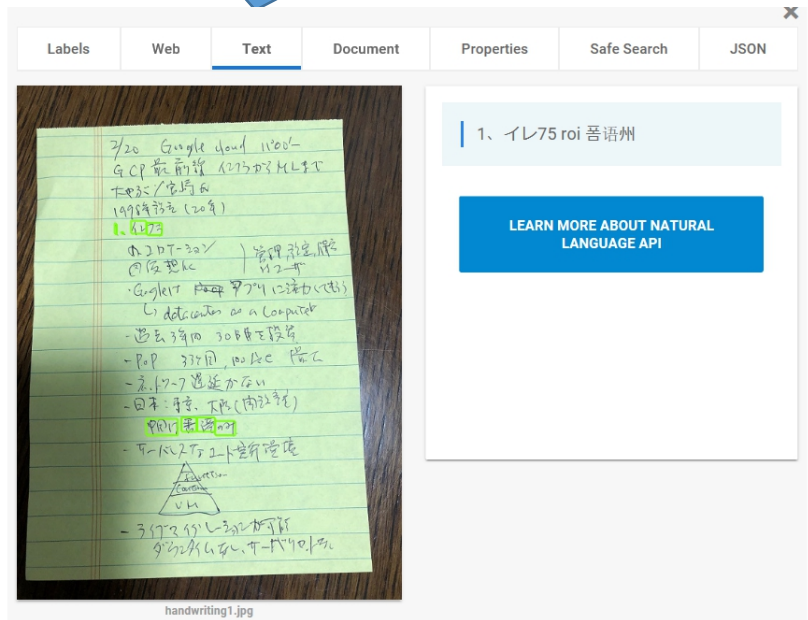
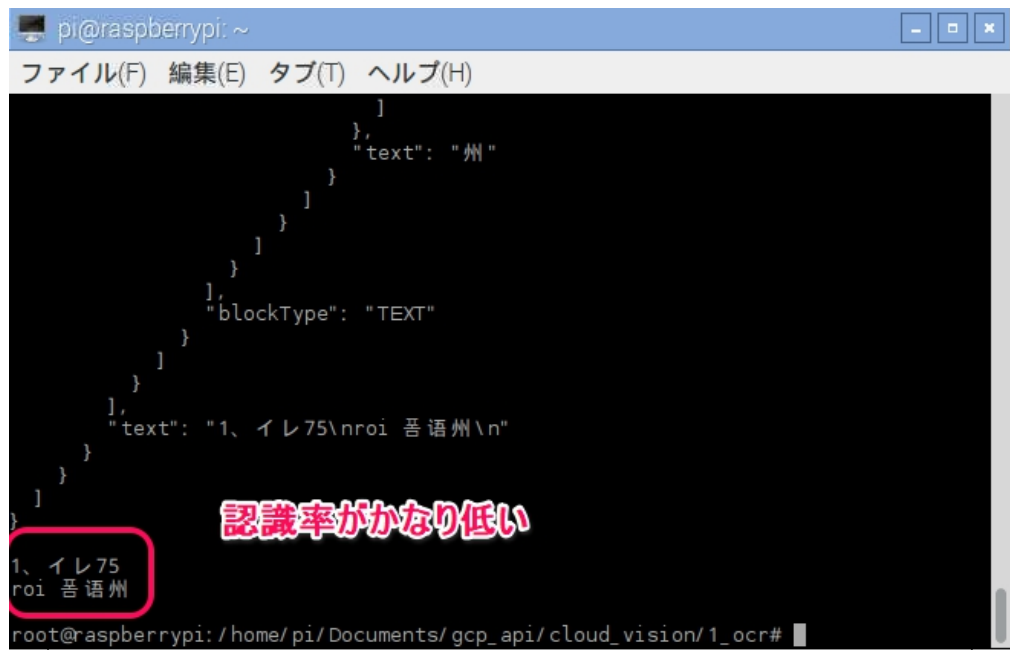
① Cloud vision>1_OCR(文字認識)

- テキスト認識: ocr3.py(手書き文字読み取り)
 - 同様にApi キーを設定します。
 - Raspberry piからocr3.pyを実行します。
 - 出力は、piの画面上とファイルのdata3.jsonの双方にです。ほとんど認識しません。
 - loadjson.pyで読み出しできます。

コマンド

```
# cd /home/pi/Documents/gcp_api/cloud_vision/1_ocr  
# python ocr3.py  
# python loadjson.py
```

Webでの確認
<https://cloud.google.com/vision/?authuser=1&hl=ja>



クラウドAI開発キット

4.使用方法

① Cloud vision>1_OCR(文字認識)

- テキスト認識: ocr4.py(ライブラリ使用)
 - 13行目に画像ファイル名を設定します。
 - ocr4.pyを実行します。
 - コマンド画面にテキストが出力されます。
 - こちらのプログラムの方が簡単ですが、jsonの出力がないため、データの加工ができません。

コマンド

```
# cd /home/pi/Documents/gcp_api/cloud_vision/1_ocr
# python ocr4.py
```

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
"text": "1、イレ75\nroi 吾語州\n"
}
}
]
}
1、イレ75
roi 吾語州

root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# export GOOGLE_APPLICATION_CREDENTIALS="/home/pi/Documents/gcp_api/doorbellapi-909617e1033d.json"
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# ls
choushi3.png data2.json handwriting1.jpg ocr1.py ocr3.py receipt1.jpg
data1.json data3.json loadjson.py ocr2.py ocr4.py
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# python ocr4.py
exts:
1625
bounds: ( 128, 222), ( 170, 222), ( 170, 241), ( 128, 241)
1625"
ounds: ( 130, 222), ( 170, 229), ( 168, 241), ( 128, 234)
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr#
```

出力

```
ocr4.py - sakura 2.2.0.1
ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)
1 #!/usr/bin/python
2 #coding:utf-8
3 import base64
4 import json
5 from requests import Request, Session
6
7 import argparse
8 import io
9
10 from google.cloud import vision
11 from google.cloud.vision import types
12
13 path = "choushi3.png"
14
15
16 def detect_text(path):
17     """Detects text in the file."""
18     client = vision.ImageAnnotatorClient()
19
20     with io.open(path, 'rb') as image_file:
21         content = image_file.read()
22
23     image = types.Image(content=content)
24
25     response = client.text_detection(image=image)
26     texts = response.text_annotations
27     print('Texts:')
28
29     for text in texts:
30         print('{}{}'.format(text.description))
31
32     vertices = ([('{}',{}).format(vertex.x, vertex.y)
33                 for vertex in text.bounding_poly.vertices])
34
35     print('bounds: {}'.format(','.join(vertices)))
36
37 if __name__ == '__main__':
38     detect_text(path)
39
40 EOF
```

画像ファイル名

クラウドAI開発キット

4.使用方法

① Cloud vision>3_label(ラベル検出)

- label1.py(http使用)
 - Raspberry piからlabel1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata4.jsonの双方にでます。

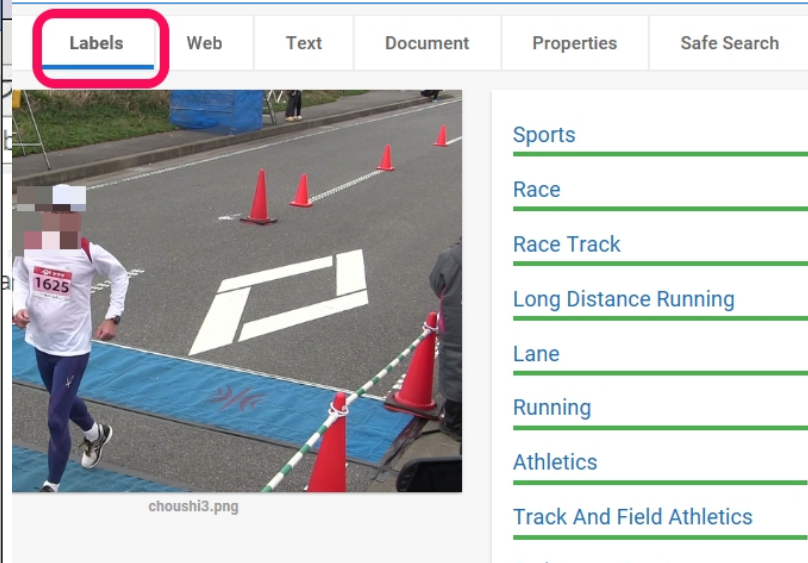
コマンド

```
# cd /home/pi/Documents/gcp_api/cloud_vision/3_label  
# python label1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
choushi3.png data4.json label1.py label2.py loadjson_l.py  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/3_label# ls  
choushi3.png data4.json label1.py label2.py loadjson_l.py  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/3_label# python label1.py  
begin request  
end request  
{  
  "responses": [  
    {  
      "labelAnnotations": [  
        {  
          "mid": "/m/06ntj",  
          "description": "sports",  
          "score": 0.9351731,  
          "topicality": 0.9351731  
        },  
        {  
          "mid": "/m/06d4",  
          "description": "race",  
          "score": 0.87084854,  
          "topicality": 0.87084854  
        }  
      ]  
    }  
  ]  
}
```



画像のカテゴリが出力されます

クラウドAI開発キット

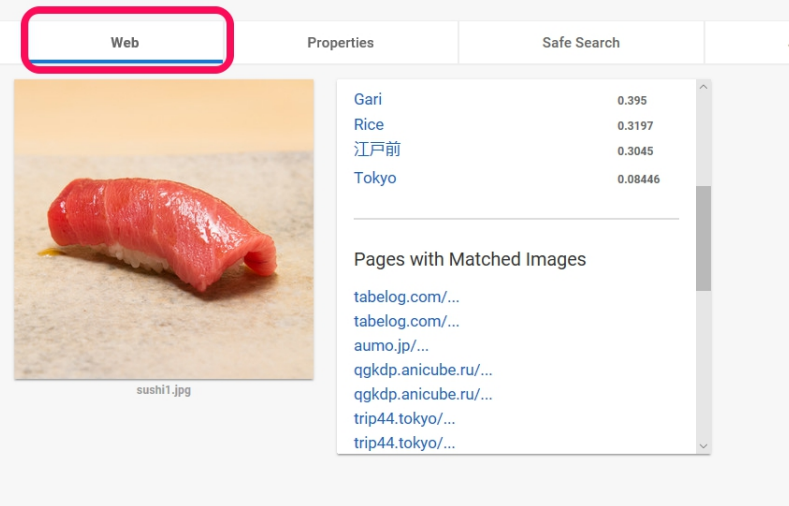
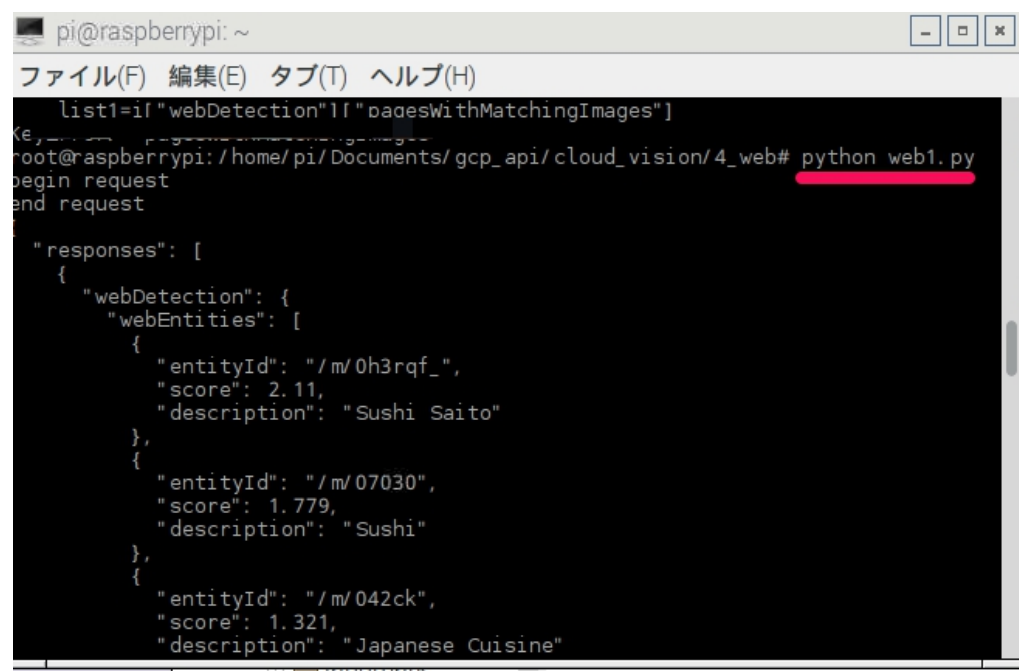
4.使用方法

① Cloud vision>4_web(web検出)

```
コマンド  
# cd /home/pi/Documents/gcp_api/cloud_vision/4_web  
# python web1.py
```

- web1.py(http使用)
 - Raspberry piからweb1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata7.jsonの双方にでます。

Webでの確認
<https://cloud.google.com/vision/?authuser=1&hl=ja>



Web上で類似した画像、URLが出力されます

クラウドAI開発キット

4.使用方法

① Cloud vision>5_safe (不適切画像検出) # cd /home/pi/Documents/gcp_api/cloud_vision/5_safe

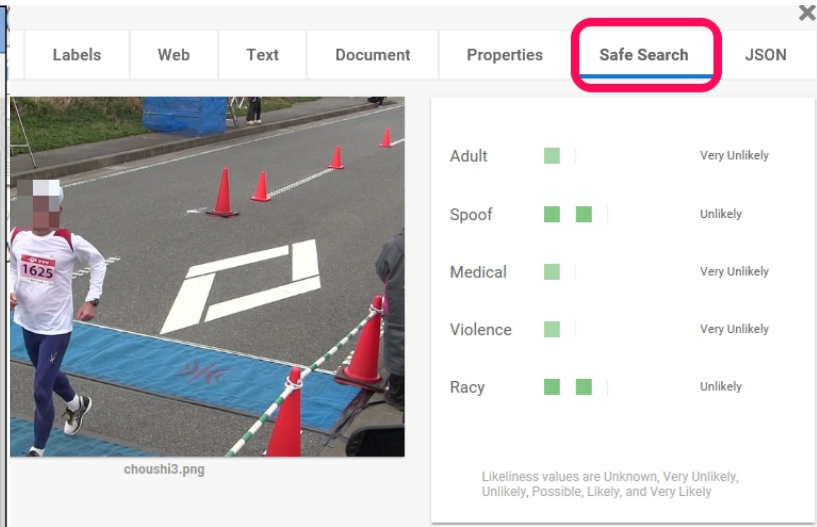
コマンド

```
# python safe1.py
```

- safe1.py (http使用)
 - Raspberry piからsafe1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata8.jsonの双方にでます。

Webでの確認
<https://cloud.google.com/vision/?authuser=1&hl=ja>

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
root@raspberrypi:~# cd /home/pi/Documents/gcp_api/cloud_vision/5_safe  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/5_safe# ls  
choushi3.png data8.json loadjson_s.py safe1.py safe2.py  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/5_safe# python safe1.py  
begin request  
end request  
  
"responses": [  
  {  
    "safeSearchAnnotation": {  
      "adult": "VERY_UNLIKELY",  
      "spooof": "UNLIKELY",  
      "medical": "VERY_UNLIKELY",  
      "violence": "VERY_UNLIKELY",  
      "racy": "UNLIKELY"  
    }  
  }  
]  
  
{u'safeSearchAnnotation': {u'medical': u'VERY_UNLIKELY', u'spooof': u'UNLIKELY',  
u'violence': u'VERY_UNLIKELY', u'adult': u'VERY_UNLIKELY', u'racy': u'UNLIKELY'  
}}
```



アダルト、暴力などの不適切画像が検出できます。

クラウドAI開発キット

4.使用方法

① Cloud vision>6_logo(ロゴ検出)

- logo1.py(http使用)
 - Raspberry piからlogo1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata9.jsonの双方にでます。

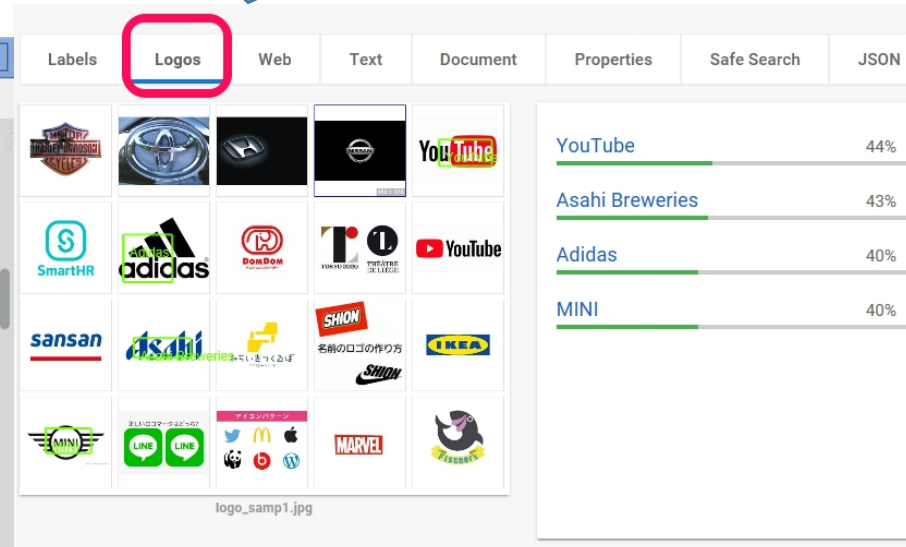
コマンド

```
# cd /home/pi/Documents/gcp_api/cloud_vision/6_logo
# python logo1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/6_logo# ls
data9.json loadjson_lo.py logo1.py logo2.py logo_samp1.jpg
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/6_logo# python logo1.py
begin request
end request
{
  "responses": [
    {
      "logoAnnotations": [
        {
          "mid": "/g/1dv8ql4m",
          "description": "YouTube",
          "score": 0.44070464,
          "boundingPoly": {
            "vertices": [
              {
                "x": 797,
                "y": 72
              },
              {
                "x": 897,
                "y": 72
              }
            ]
          }
        }
      ]
    }
  ]
}
```



ロゴ検出できます。テキストがない場合は検出できません

クラウドAI開発キット

4.使用方法

① Cloud vision>7_landmark(ランドマーク検出)

- landm1.py(http使用)
 - Raspberry piからlandm1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata11.jsonの双方にでます。

コマンド

```
# cd
```

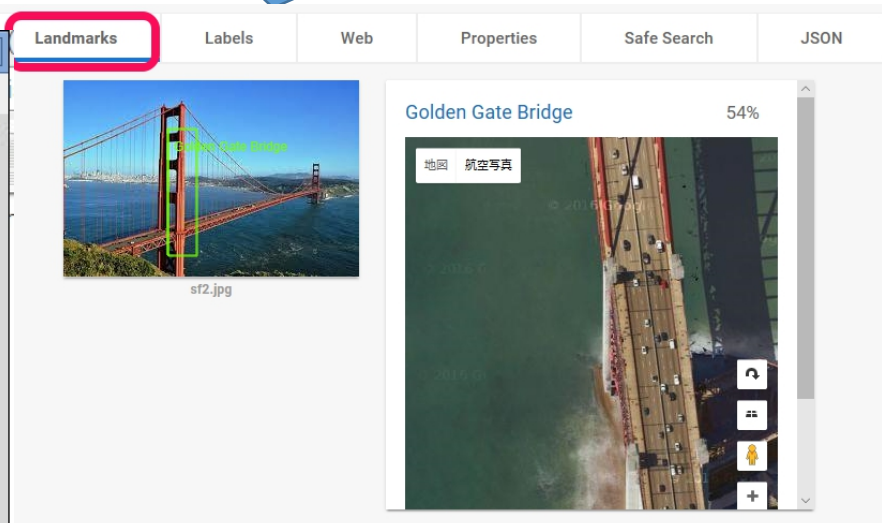
```
/home/pi/Documents/gcp_api/cloud_vision/7_landmark
```

```
# python landm1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/7_landmark# python landm1.py  
begin request  
end request  
{  
  "responses": [  
    {  
      "landmarkAnnotations": [  
        {  
          "mid": "/m/035p3",  
          "description": "Golden Gate Bridge",  
          "score": 0.54256004,  
          "boundingPoly": {  
            "vertices": [  
              {  
                "x": 98,  
                "y": 46  
              },  
              {  
                "x": 125,  
                "y": 46  
              },  
              {  
                "x": 125,  
                "y": 125  
              }  
            ]  
          }  
        }  
      ]  
    }  
  ]  
}
```



ランドマークが検出できます。

クラウドAI開発キット

4.使用方法

① Cloud vision>8_prop(プロパティ検出)

- Prop1.py(http使用)
 - Raspberry piからprop1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata10.jsonの双方にでます。

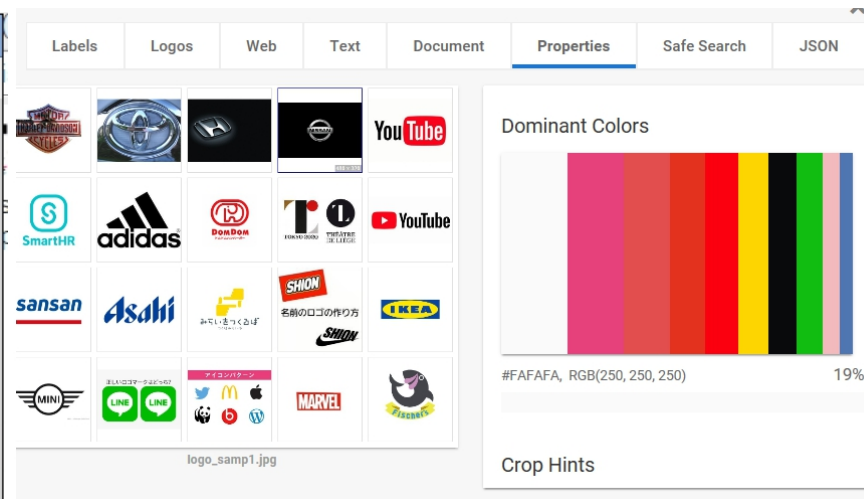
コマンド

```
# cd  
/home/pi/Documents/gcp_api/cloud_vision/8_prop  
# python prop1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/8_prop# ls  
data10.json loadjson_p.py logo_samp1.jpg prop1.py prop2.py  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/8_prop# python prop1.py  
begin request  
end request  
{  
  "responses": [  
    {  
      "imagePropertiesAnnotation": {  
        "dominantColors": {  
          "colors": [  
            {  
              "color": {  
                "red": 250,  
                "green": 250,  
                "blue": 250  
              },  
              "score": 0.11050146,  
              "pixelFraction": 0.7340543  
            },  
            {  
              "color": {  
                "red": 231,  
                "green": 65,  
                "blue": 250  
              },  
              "score": 0.08949854,  
              "pixelFraction": 0.2659457  
            }  
          ]  
        }  
      }  
    }  
  ]  
}
```



画像の色などのプロパティ情報が検出できます。

クラウドAI開発キット

4.使用方法

① Cloud vision>8_prop(プロパティ検出)

- Prop2.py(ライブラリ使用)
 - サービスアカウントキーの設定が完了していること。16ページ参照
 - 13行目に画像ファイル名を設定します。
 - prop2.pyを実行します。
 - コマンド画面にテキストが出力されます。
 - こちらのプログラムの方が簡単ですが、出力データが少ないです

```
コマンド
# cd
/home/pi/Documents/gcp_api/cloud_vision/8_prop
# python prop2.py
```

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
({u'blue': 177, u'green': 118, u'red': 79})
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/8_prop# python prop2.py
Properties:
fraction: 0.734054327011
r: 250.0
g: 250.0
b: 250.0
a:
fraction: 0.00377928954549
r: 231.0
g: 65.0
b: 123.0
a:
fraction: 0.00343043194152
r: 226.0
g: 78.0
b: 78.0
a:
fraction: 0.00354671780951
r: 227.0
g: 50.0
b: 29.0
a:
fraction: 0.00331414607354
```

```
デスクトップ#murakami_jeno#ビジネス#技術関係#deep_learning#gcp_ai#cloud_vision#8_prop#prop2.py - sakura 2.2.0.1
ファイル(F) 編集(E) 変換(O) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)
1 #!/usr/bin/python
2 #coding:utf-8
3 import base64
4 import json
5 from requests import Request, Session
6
7 import argparse
8 import io
9
10 from google.cloud import vision
11 from google.cloud.vision import types
12
13 path = "logo_sampl.jpg"
14
15 def detect_properties(path):
16     """Detects image properties in the file."""
17     client = vision.ImageAnnotatorClient()
18
19     with io.open(path, 'rb') as image_file:
20         content = image_file.read()
21
22     image = types.Image(content=content)
23
24     response = client.image_properties(image=image)
25     props = response.image_properties_annotation
26     print('Properties:')
27
28     for color in props.dominant_colors.colors:
29         print('fraction: {}'.format(color.pixel_fraction))
30         print('%tr: {}'.format(color.color.red))
31         print('%tg: {}'.format(color.color.green))
32         print('%tb: {}'.format(color.color.blue))
33         print('%ta: {}'.format(color.color.alpha))
34
35 if __name__ == '__main__':
36     detect_properties(path)
37
38 EOF
```


クラウドAI開発キット

4.使用方法

① Cloud vision>9_crop(切り出し検出)

- Crop2.py(ライブラリ使用)
 - Raspberry piからpython crop2.py choushi3.jpg cropを実行します。
 - 画像ファイルは、crop2.pyの後に設定します。Jpgのみ有効です。
 - 出力は、output-crop.jpgです。

```
コマンド  
# cd  
/home/pi/Documents/gcp_api/cloud_vision/9_crop  
# python crop2.py choushi3.jpg crop
```



choushi3.jpg



output-crop.jpg

クラウドAI開発キット

4.使用方法

② Cloud speech>transcribe_stream_mic(マイク音声認識)

- transcribe_streaming_mic.py(マイク音声認識)
 - transcribe_streaming_mic.pyを実行します。
 - マイクにしゃべると自動で、テキストが出力されます。すごいです。

コマンド

```
# cd  
/home/pi/Documents/gcp_api/cloud  
_speech/  
# python  
transcribe_streaming_mic.py
```

音楽ファイルは、バックグラウンドに音ははいているため、うまく出力できません。

```
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_speech# ls  
output.wav      transcribe.py      transcribe_streaming_mic.py  
pyaudio_test.py transcribe_async.py  
test030701.wav  transcribe_streaming.py  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_speech# python transcribe_streaming_mic.py  
ALSA lib confmisc.c:1286:(snd_func_refer) Unable to find definition 'cards.bcm2835.pcm.front.0:CARD=0'  
ALSA lib conf.c:4259:(snd_config_evaluate) function snd_func_refer returned error: no such file or directory  
ALSA lib conf.c:4338:(snd_config_expand) Fatal error: Invalid configuration
```

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
ALSA lib pcm.c:2239:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.modem  
ALSA lib pcm.c:2239:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline  
ALSA lib pcm.c:2239:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline  
ALSA lib pulse.c:243:(pulse_connect) PulseAudio: Unable to connect: Connection refused  
  
ALSA lib pulse.c:243:(pulse_connect) PulseAudio: Unable to connect: Connection refused  
  
Cannot connect to server socket err = No such file or directory  
Cannot connect to server request channel  
jack server is not running or cannot be started  
今日は晴天なり今日は晴天なりテストテスト今日もいい天気です J WAVE 聞いてま  
^C  
Traceback (most recent call last):  
  File "transcribe_streaming_mic.py", line 195, in <module>  
    main()  
  File "transcribe_streaming_mic.py", line 191, in main  
    listen_print_loop(responses)  
  File "transcribe_streaming_mic.py", line 129, in listen_print_loop  
    for response in responses:  
  File "/usr/local/lib/python2.7/dist-packages/grpc/_channel.py", line 350, in next  
    all rights reserved 2013 spectrum technology co.  
  File "/usr/local/lib/python2.7/dist-packages/google/api_core/grpc_helpers.py",
```

```
transcribe_streaming_mic.py - sakura 2.2.0.1  
ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)  
19  
20 NOTE: This module requires the additional dependency `pyaudio`. To install  
21 using pip:  
22  
23     pip install pyaudio  
24  
25 Example usage:  
26     python transcribe_streaming_mic.py  
27  
28  
29 # [START import_libraries]  
30 from __future__ import division  
31  
32 import re  
33 import sys  
34  
35 from google.cloud import speech  
36 from google.cloud.speech import enums  
37 from google.cloud.speech import types  
38 import pyaudio  
39 from six.moves import queue  
40 # [END import_libraries]  
41  
42 # Audio recording parameters  
43 RATE = 16000  
44 CHUNK = int(RATE / 10) # 100ms  
45  
46  
47 class MicrophoneStream(object):  
48     """Opens a recording stream as a generator yielding the audio chunks."""  
49     def __init__(self, rate, chunk):  
50         self.rate = rate  
51         self.chunk = chunk  
52  
53         # Create a thread-safe buffer of audio data  
54         self._buff = queue.Queue()  
55         self.closed = True  
56  
57     def __enter__(self):  
58         self.audio_interface = pyaudio.PyAudio()  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

クラウドAI開発キット

4.使用方法

③ Cloud nl>sentiment(感情分析)

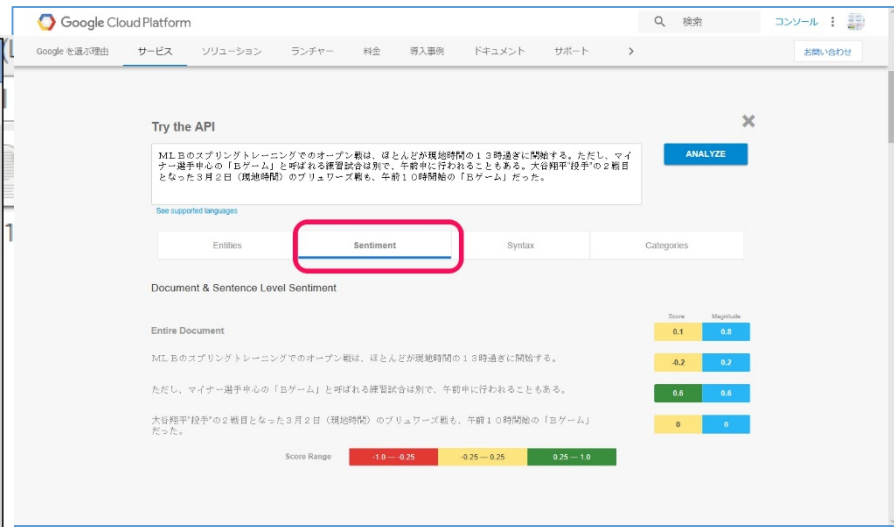
- senti2.py(文節解析)
 - senti2.pyを実行します。文節毎に値を出します。
 - テスト文はプログラムの後ろに指定します。
 - スコア(ポジティブ、ニュートラル、ネガティブ)と感情の強度がでます。詳細は以下のURL参照。
 - https://cloud.google.com/natural-language/docs/basics?authuser=1&hl=ja#interpreting_sentiment_analysis_values

コマンド

```
# cd /home/pi/Documents/gcp_api/cloud_nl/ # python senti2.py test_sentence.txt
```

Webでの確認
<https://cloud.google.com/natural-language/?authuser=1&hl=ja>

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
File "/usr/local/lib/python2.7/dist-packages/google/api_core/grpc_helpers.py", line 54, in error_remapped_callable
    return callable(*args, **kwargs)
File "/usr/local/lib/python2.7/dist-packages/grpc/_channel.py", line 332, in next
    self._state.condition.wait()
File "/usr/lib/python2.7/threading.py", line 340, in wait
    waiter.acquire()
KeyboardInterrupt
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_speech# cd /home/pi/Documents/gcp_api/cloud_nl
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_nl# ls
classify1.py hello_world.py senti2.py test_sentence.txt
entity1.py senti1.py syntax1.py test_sentence_en.txt
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_nl# python senti1.py
Score: 0.10000000149
Magnitude: 0.800000011921
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_nl# python senti2.py test_sentence.txt
sentence 0 has a sentiment score of -0.20000000298
sentence 1 has a sentiment score of 0.600000023842
sentence 2 has a sentiment score of 0.0
Overall Sentiment: score of 0.10000000149 with magnitude of 0.800000011921
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_nl#
```



クラウドAI開発キット

4.使用方法

③ Cloud nl>syntax(構文解析)

- syntax1.py(構文解析)
 - Syntax1.pyを実行します。単語ごとに解析します。名詞、接続詞、副詞、動詞など。
 - 14行目にテキストのファイル名を設定しています。test_sentence.txtを添付しています。

```

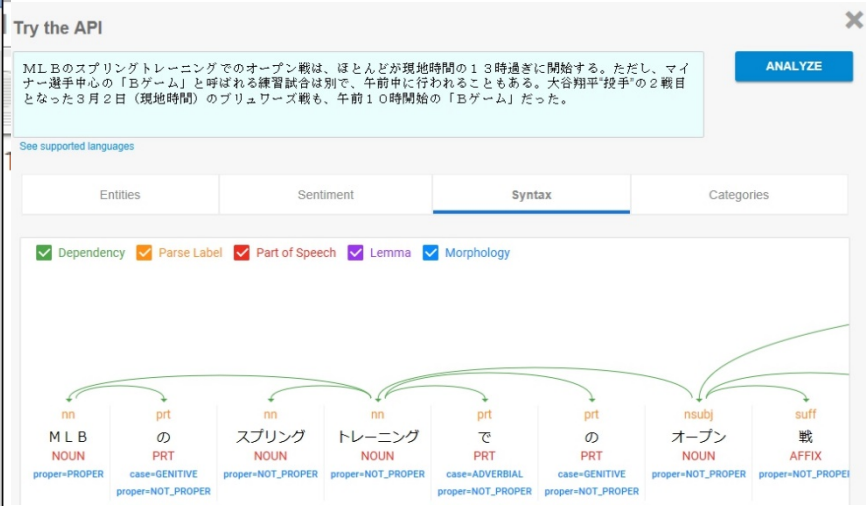
コマンド
# cd
/home/pi/Documents/gcp_api/cloud_nl/
# python syntax1.py
    
```

Webでの確認
<https://cloud.google.com/natural-language/?authuser=1&hl=ja>

```

pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
Overall Sentiment: score of 0.10000000149 with magnitude of 0.800000011921
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_nl# python syntax1.py
NOUN: M L
PRT: の
NOUN: のスプリ
NOUN: グトレーニング
PRT: の
PRT: で
NOUN: のオーブ
AFFIX: ン
PRT: 戦
PUNCT: は
NOUN: 、ほとん
PRT: ど
NOUN: が現
NOUN: 地時
PRT: 間
NUM: の1
AFFIX: 3
AFFIX: 時過
PRT: ぎ
NOUN: に関
VERB: 始す
PUNCT: る
    
```

1文字づつず
 れています。原
 因不明



クラウドAI開発キット

4.使用方法

③ Cloud nl>entity(エンティティ分析)

- entity1.py(エンティティ分析)
 - entity1.pyを実行します。人物、場所、組織、イベント、URLなどを分析します。
 - 14行目にテキストのファイル名を設定しています。test_sentence.txtを添付しています。

コマンド

```
# cd
/home/pi/Documents/gcp_api/cloud_nl/
# python entity1.py
```

Webでの確認

<https://cloud.google.com/natural-language/?authuser=1&hl=ja>

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
entity1.py senti2.py test_sentence.txt
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_nl# python entity1.py
=====
name      : 現地時間
type      : OTHER
metadata  : {}
salience : 0.163908243179
wikipedia_url : -
=====
name      : Bゲーム
type      : CONSUMER_GOOD
metadata  : {}
salience : 0.130278632045
wikipedia_url : -
=====
name      : MLB
type      : ORGANIZATION
metadata  : {'mid': ' /m/09p14', 'wikipedia_url': ' https://en.wikipedia.org/wiki/Major_League_Baseball'}
salience : 0.105740621686
wikipedia_url : https://en.wikipedia.org/wiki/Major_League_Baseball
=====
name      : スプリングトレーニング
type      : EVENT
```

Try the API

MLBのスプリングトレーニングでのオープン戦は、ほとんどが現地時間の13時過ぎに開始する。ただし、マイナー選手中心の「Bゲーム」と呼ばれる練習試合は別で、午前中に行われることもある。大谷翔平投手の2戦目となった3月2日(現地時間)のブルージェイズ戦も、午前10時開始の「Bゲーム」だった。

See supported languages

Entities	Sentiment	Syntax	Categories
----------	-----------	--------	------------

(MLB)₃の(スプリングトレーニング)₄での(オープン戦)₅は、(ほとんど)₆が(現地時間)₁の13時過ぎに開始する。ただし、(マイナー選手)₁₁(Bゲーム)₂と呼ばれる(練習試合)は(別)で、午前中に行われる(こと)₁₄もある。(大谷翔平)₁₅(投手)₁₆の2戦目となった3月2日((現地時間)ブルージェイズ)₁₃(戦)₇も、午前10時(開始)₁₀の「(Bゲーム)₂」だった。

1. 現地時間 Salience: 0.16	OTHER	2. Bゲーム Salience: 0.13	CONSUMER_GOOD
3. MLB Wikipedia Article Salience: 0.11	ORGANIZATION	4. スプリングトレーニング Salience: 0.10	
5. オープン戦 Salience: 0.09	EVENT	6. ほとんど Salience: 0.08	

クラウドAI開発キット

4.使用方法

④ Cloud translate (翻訳)

- Quickstart.py (翻訳)
 - quickstart.pyを実行します。
 - 18行目にテキストのファイル名を設定しています。test_sentence_en.txtを添付しています。
 - 翻訳先言語名は、
<https://cloud.google.com/translate/docs/languages?authuser=1&hl=ja>

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_translate# ls  
quickstart.py snippets.py test_sentence_en.txt  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_translate# python quickstart.py  
Text: President Donald Trump signaled that he's open to talks with North Korea after Kim Jong Un's regime told South Korean envoys that he's willing to consider giving up his nuclear weapons ? a potential breakthrough after months of bellicose threats from both leaders.  
"They seem to be acting positively," Trump told reporters Tuesday. "I'd like to be optimistic."  
Trump commented after envoys from Seoul said that Kim told them he was ready to suspend weapons tests and hold candid talks with the U.S. to normalize relations, if the safety of his regime was guaranteed, the South Korean government said Tuesday. In response, South Korean President Moon Jae-in agreed to meet Kim for a summit along their shared border late next month.  
Translation: ドナルド・トランプ大統領は、金正日(キム・ジョンウン)政権が韓国の使節に対し、核兵器を諦めることを喜んでいると語った後、彼は北朝鮮との交渉を開いていると伝えた。両方の指導者から数ヶ月にわたる絶滅の危機に突入する可能性があります。&quot;彼らは積極的に行動しているようだ&quot;とトランプは火曜日に記者団に語った。&quot;私は楽観的になってほしい&quot;とトランプはコメントした。ソウル出身の大使は、金総書記が武器検査を中断し、米国との関係を正常化する用意があると述べ、韓国政府は火曜日に言った。これに対して韓国のムン・ジェイン(ムン・ジェイン)大統領は金次官補と来月下旬に国境に沿って首脳会談を行うことで合意した。  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_translate#
```

```
コマンド  
# cd  
/home/pi/Documents/gcp_api/cloud_translate/  
# python quickstart.py
```

```
デスクトップ#murakami_jeno#ビジネス#技術関係#deep_learning#gcp_api#cloud_translate#quickstart#quickstart.py - saku...  
ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)  
1 #!/usr/bin/env python+  
2 #coding:utf-8+  
3  
4 # Copyright 2016 Google Inc. All Rights Reserved.+  
5  
6 # Licensed under the Apache License, Version 2.0 (the "License");+  
7 # you may not use this file except in compliance with the License.+  
8 # You may obtain a copy of the License at+  
9 #+  
10 # http://www.apache.org/licenses/LICENSE-2.0+  
11 #+  
12 # Unless required by applicable law or agreed to in writing, software+  
13 # distributed under the License is distributed on an "AS IS" BASIS,+  
14 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.+  
15 # See the License for the specific language governing permissions and+  
16 # limitations under the License.+  
17 import six+  
18 path = "test_sentence_en.txt" ← 対象テキストファイル名  
19  
20 def run_quickstart():+  
21 # [START translate_quickstart]+  
22 # Imports the Google Cloud client library+  
23 from google.cloud import translate+  
24  
25 # Instantiates a client+  
26 translate_client = translate.Client()+  
27  
28 # The text to translate+  
29 text=open(path, "rb").read()+  
30 if isinstance(text, six.binary_type):+  
31 text = text.decode("utf-8")+  
32 #text = u'Hello, world!'+  
33 # The target language+  
34 target = 'ja' ← 日本語への翻訳  
35  
36 # Translates some text into Russian+  
37 translation = translate_client.translate(+  
38 text,+  
39 target_language=target)+  
40
```

クラウドAI開発キット

4.使用方法

⑤ Cloud video

- label1.py(ラベル分析)
 - label1.pyを実行します。ローカルの動画を分析します。
 - 11行目にファイル名を記載しています。
 - 出力は、動画分析するため数分かかります。
 - 関連するカテゴリが時間別に出力されます。

```
コマンド
# cd
/home/pi/Documents/gcp_api/cloud_video/
# python label1.py
```

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_video# ls
adult1.py analyze.py aws_pr1.mp4 label1.py
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_video# python label1.py
Processing video for label annotations:
Finished processing.
Video label description: document
Label category description: paper
Segment 0: 0.0s to 143.743743s
Confidence: 0.550543308258

Video label description: website
Label category description: weak entity
Segment 0: 0.0s to 143.743743s
Confidence: 0.571462094784

Video label description: online advertising
Label category description: advertising
Segment 0: 0.0s to 143.743743s
Confidence: 0.485180169344
```

```
デスクトップ#murakami_jeno#ビジネス#技術関係#deep_learning#gcp_ai#cloud_video#label#label1.py - sakura 2.2.0.1
ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)
1 #!/usr/bin/python
2 #coding:utf-8
3 import base64
4 import json
5 import codecs
6 from requests import Request, Session
7
8 from google.cloud import videointelligence
9 import io
10
11 path = "aws_pr1.mp4"
12
13
14 def analyze_labels_file(path):
15     """Detect labels given a file path."""
16     video_client = videointelligence.VideoIntelligenceServiceClient()
17     features = [videointelligence.enums.Feature.LABEL_DETECTION]
18
19     with io.open(path, 'rb') as movie:
20         input_content = movie.read()
21
22     operation = video_client.annotate_video(
23         features=features, input_content=input_content)
24     print('\nProcessing video for label annotations:')
25
26     result = operation.result(timeout=90)
27     print('\nFinished processing.')
28
29     # Process video/segment level label annotations
30     segment_labels = result.annotation_results[0].segment_label_annotations
31     for i, segment_label in enumerate(segment_labels):
32         print("Video label description: {}".format(
33             segment_label.entity.description))
34         for category_entity in segment_label.category_entities:
35             print("%fLabel category description: {}".format(
36                 category_entity.description))
37
38     for i, segment in enumerate(segment_label.segments):
39         start_time = (segment.segment.start_time_offset.seconds +
40                      segment.segment.start_time_offset.nanos / 1e9)
```


クラウドAI開発キット

4.使用方法

⑤ Cloud video

- adult1.py (不適切動画認識)
 - adult1.pyを実行します。ローカルの動画を分析します。
 - 11行目にファイル名を記載しています。
 - 出力は、動画分析するため数分かかります。
 - 関連するカテゴリが時間別に出力されます。

```
コマンド  
# cd  
/home/pi/Documents/gcp_api/cloud_video/  
# python adult1.py
```

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_video# ls  
adult1.py analyze.py aws_pr1.mp4 label1.py  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_video# python adult1.py  
Processing video for explicit content annotations:  
Finished processing.  
Time: 0.594403s  
pornography: Very unlikely  
Time: 1.719093s  
pornography: Very unlikely  
Time: 2.575706s  
pornography: Very unlikely  
Time: 3.616114s  
pornography: Very unlikely  
Time: 4.798984s  
pornography: Very unlikely  
Time: 5.694583s  
pornography: Very unlikely  
Time: 6.571351s  
pornography: Very unlikely  
Time: 7.46716s  
pornography: Very unlikely
```

```
デスクトップmurakami_ieno%ビジネス%技術関係%deep_learning%gcp_ai%cloud_video%adult%adult1.py - sakura 2.2.0.1  
ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)  
1 #!/usr/bin/python  
2 #coding:utf-8  
3 import base64  
4 import json  
5 import codecs  
6 from requests import Request, Session  
7  
8 from google.cloud import videointelligence  
9 import io  
10  
11 path = "aws_pr1.mp4"  
12  
13 def analyze_explicit_content(path):  
14     """ Detects explicit content from the GCS path to a video. """  
15     video_client = videointelligence.VideoIntelligenceServiceClient()  
16     features = [videointelligence.enums.Feature.EXPLICIT_CONTENT_DETECTION]  
17  
18     with io.open(path, 'rb') as movie:  
19         input_content = movie.read()  
20  
21     operation = video_client.annotate_video(  
22         features=features, input_content=input_content)  
23     #operation = video_client.annotate_video(path, features=features)  
24     print("\nProcessing video for explicit content annotations:")  
25  
26     result = operation.result(timeout=90)  
27     print("\nFinished processing.")  
28  
29     likely_string = ("Unknown", "Very unlikely", "Unlikely", "Possible",  
30                     "Likely", "Very likely")  
31  
32     # first result is retrieved because a single video was processed  
33     for frame in result.annotation_results[0].explicit_annotation.frames:  
34         frame_time = frame.time_offset.seconds + frame.time_offset.nanos / 1e9  
35         print("Time: {}s".format(frame_time))  
36         print("%tpornography: {}".format(  
37             likely_string[frame.pornography_likelihood]))  
38  
39 if __name__ == '__main__':  
40     analyze_explicit_content(path)
```