

Huawei HCIP-AI Certification Training

HCIP-AI EI Developer

Image Processing

Experiment Guide

Issue: 1.0



HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Copyright © Huawei Technologies Co., Ltd. 2019. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <http://e.huawei.com>

Huawei Certificate System

Huawei Certification follows the "platform + ecosystem" development strategy, which is a new collaborative architecture of ICT infrastructure based on "Cloud-Pipe-Terminal". Huawei has set up a complete certification system consisting of three categories: ICT infrastructure certification, Platform and Service certification and ICT vertical certification, and grants Huawei certification the only all-range technical certification in the industry.

Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

The fourth revolution of science and technology, driven by ICT networks and AI, is leading us towards an intelligent world where all things can sense, all things are connected, and all things are intelligent. In October 2018, Huawei Rotating Chairman Eric Xu announced Huawei's AI strategy, as well as its full-stack, all-scenario AI portfolio. The company's full-stack, all-scenario AI portfolio is designed to provide powerful support for Huawei Cloud EI and HiAI. Backed by its AI portfolio, Huawei Cloud EI will be able to deliver a full-stack portfolio for enterprise and government customers, and HiAI will provide a full-stack portfolio for smart devices. HiAI services are deployed on Huawei Cloud EI. However, the gap of AI talent in this world is very large. To seize this historic opportunity and accelerate the construction of the AI talent ecosystem, we hereby release the HCIP-AI EI Developer V1.0 certification to develop talents who have acquired basic theoretical knowledge about image processing, speech processing, and natural language processing (NLP) and who are able to use Huawei enterprise AI solutions (such as Huawei cloud EI) and general open-source frameworks to conduct development and innovation.

HCIP-AI EI Developer V1.0 is a kind of engineer career certification for ICT practitioners, Huawei's users, Huawei engineers, Huawei partners' engineers, and college students. The content of HCIP-AI EI Developer V1.0 includes but is not limited to: deep feedforward networks, image processing theory and applications, speech processing theory and applications, NLP theory and applications, the image processing experiment, the speech processing experiment, and the NLP experiment.

About This Document

Overview

This document is an HCIP-AI certification training course for trainees who are preparing for HCIP-AI exams or readers who want to know about AI basics. Through learning of this course, you will understand how to preprocess image files and use HUAWEI CLOUD to develop functions including face recognition, image tagging, optical character recognition, and image content moderation. The OpenCV library is used to implement the image preprocessing experiment. RESTful requests are submitted to invoke HUAWEI CLOUD services to implement the face recognition, image tagging, optical character recognition, and content moderation experiments. HUAWEI CLOUD Enterprise Intelligence (EI) provides various APIs for image processing applications.

Description

This document contains five experiments, involving the OpenCV image preprocessing experiment based on the Python language, and experiments of face recognition, image tagging, optical character recognition, and content moderation based on the Java language and HUAWEI CLOUD. The purpose of this document is to improve the development capability related to AI image processing.

- Experiment 1: OpenCV-based Image Preprocessing
- Experiment 2: Image Tagging Based on Java and HUAWEI CLOUD
- Experiment 3: Content Moderation Based on Java and HUAWEI CLOUD
- Experiment 4: Optical Character Recognition Based on Java and HUAWEI CLOUD
- Experiment 5: Face Recognition Based on Java and HUAWEI CLOUD

Background Knowledge Required

This course is designed for development training for Huawei certification. To better understand content of this document, readers are required to meet the following basic conditions:

- Have basic language editing capability
- Have basic knowledge in data structure
- Have basic knowledge in Java RESTful programming

Experiment Environment Overview

- Python 3.6, OpenCV, NumPy, Matplotlib
- JDK 1.8, Maven 3.4.9 or later
- Eclipse 4.6

Contents

About This Document	1
Overview	2
Description	2
Background Knowledge Required	3
Experiment Environment Overview	3
1 Image Data Preprocessing.....	7
1.1 Experiment Introduction.....	7
1.2 Experiment Objectives	7
1.3 Experiment Environment Overview	7
1.4 Procedure	8
1.4.1 Basic Operations.....	8
1.4.2 Color Space Conversion.....	8
1.4.3 Coordinate Transformation	9
1.4.4 Grayscale Transformation	12
1.4.5 Histogram.....	13
1.4.6 Filtering.....	14
1.5 Summary	17
2 Image Recognition	18
2.1 Experiment Introduction.....	18
2.2 Experiment Objectives	18
2.3 API Introduction	19
2.3.1 RESTful API Introduction.....	19
2.3.2 RESTful API Request/Response Composition.....	19
2.3.3 Request Initiation Methods.....	21
2.3.4 Image Tagging API	21
2.4 Procedure	23
2.4.1 Applying for the Service	24

2.4.2 Downloading the Image Recognition SDK	24
2.4.3 Loading the SDK Project File	27
2.4.4 Using the Image Tagging Service Through the AK/SK	28
2.5 Summary	32
3 Content Moderation	33
3.1 Experiment Introduction	33
3.2 Experiment Objectives	33
3.3 API Introduction	34
3.3.1 RESTful API Introduction	34
3.3.2 RESTful API Request/Response Composition	34
3.3.3 Request Initiation Methods	36
3.3.4 Image Content Moderation API	36
3.4 Procedure	41
3.4.1 Applying for the Service	41
3.4.2 Downloading the Image Content Moderation SDK	41
3.4.3 Loading the SDK Project File	44
3.4.4 Using the Image Content Moderation Service Through the AK/SK	45
3.5 Summary	51
4 OCR	52
4.1 Experiment Introduction	52
4.2 Experiment Objectives	52
4.3 API Introduction	53
4.3.1 RESTful API Introduction	53
4.3.2 RESTful API Request/Response Composition	53
4.3.3 General Text OCR API	54
4.3.4 ID Card OCR API	57
4.3.5 VAT Invoice OCR API	60
4.4 Procedure	63
4.4.1 Applying for the Service	63
4.4.2 Downloading the OCR SDK	64
4.4.3 Loading the SDK Project File	66

4.4.4 Using the General Text OCR Service Through the AK/SK	67
4.4.5 Using the ID Card OCR Service Through the AK/SK	72
4.4.6 Using the VAT Invoice OCR Service Through the AK/SK.....	76
4.5 Summary	81
5 Face Recognition.....	82
5.1 Experiment Introduction.....	82
5.2 Experiment Objectives	82
5.3 API Introduction	83
5.3.1 RESTful API Introduction.....	83
5.3.2 RESTful API Request/Response Composition.....	83
5.3.3 Request Initiation Methods.....	85
5.3.4 Obtaining the Project ID	85
5.3.5 Face Detection API	86
5.3.6 Face Verification API.....	88
5.4 Procedure	92
5.4.1 Applying for the Service	92
5.4.2 Using the Face Detection Service Through the AK/SK.....	92
5.4.3 Using the Face Verification Service Through the AK/SK	96
5.5 Summary	99

1 Image Data Preprocessing

1.1 Experiment Introduction

The main purpose of image preprocessing is to eliminate irrelevant information in images, restore useful information, enhance information detectability, and simplify data to the maximum extent, thus improving the reliability of feature extraction and image segmentation, matching, and recognition.

In this experiment, the OpenCV image processing library is used to implement basic image preprocessing operations, including color space conversion, coordinate transformation, grayscale transformation, histogram transformation, and image filtering.

1.2 Experiment Objectives

In this experiment, the image preprocessing technology introduced in the theoretical teaching materials is implemented through the OpenCV image processing library in Python. Through this experiment, trainees will learn how to use the OpenCV to preprocess images. During the experiments, trainees can observe the changes in image data to better understand the image preprocessing technology. This course helps trainees understand and master how to use Python to preprocess images.

1.3 Experiment Environment Overview

You are advised to install Python 3.6 or later, and install the OpenCV, NumPy, and Matplotlib third-party libraries.

1.4 Procedure

1.4.1 Basic Operations

Step 1 Read and display an image.

```
import cv2
# Read an image. The first argument indicates the image path.
# The second argument indicates the read mode. The value 1 indicates color
(three-channel), and the value 0 indicates grayscale (one channel).
im = cv2.imread(r"lena512color.tiff",1)
# Display the image im in the test window.
cv2.imshow("test",im)
# Wait for a user to press a key for feedback.
cv2.waitKey()
# Destroy all created windows.
cv2.destroyAllWindows()
```

Step 2 View the data type and image size.

```
#Display the data type of the image.
print(type(im))
# Display the image size.
print(im.shape)
```

Step 3 Save the image.

```
# Save the image to the specified path.
cv2.imwrite('lena.jpg',im)
```

----End

1.4.2 Color Space Conversion

Step 1 Convert a colorful image into a grayscale one.

```
import cv2
im = cv2.imread(r"lena.jpg")
cv2.imshow("BGR",im)
# Use cvtColor to perform color space conversion. The flag cv2.COLOR_BGR2GRAY
indicates BGR to gray.
img_gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray",img_gray)
```

```
cv2.waitKey()  
cv2.destroyAllWindows()
```

Step 2 Change the three-channel sequence from BGR to RGB.

```
import cv2  
im = cv2.imread(r"lena.jpg")  
cv2.imshow("BGR",im)  
# Use cvtColor to perform color space conversion. The flag cv2.COLOR_BGR2RGB  
indicates BGR to RGB.  
im_rgb = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)  
# For a three-channel image, the imshow function considers that the data is  
of the BGR type.  
# Run the imshow command to display data of the RGB type. The colors of the  
image are distorted.  
cv2.imshow("RGB",im_rgb)  
cv2.waitKey()  
cv2.destroyAllWindows()
```

Step 3 Perform color space conversion between BGR and HSV.

```
import cv2  
im = cv2.imread(r"lena.jpg")  
cv2.imshow("BGR",im)  
# Use cvtColor to perform color space conversion. The flag cv2.COLOR_BGR2HSV  
indicates BGR to HSV.  
im_hsv = cv2.cvtColor(im, cv2.COLOR_BGR2HSV)  
# For a three-channel image, the imshow function considers that the data is  
of the BGR type.  
# Run the imshow command to display HSV data. The HSV component is forcibly  
displayed as BGR.  
cv2.imshow("HSV",im_hsv)  
cv2.waitKey()  
cv2.destroyAllWindows()
```

----End

1.4.3 Coordinate Transformation

Step 1 Perform translation.

```
import numpy as np  
import cv2  
# Define the translate function.  
def translate(img, x, y):  
    # Obtain the image size.  
    (h, w) = img.shape[:2]
```

```
# Define the translation matrix.
M = np.float32([[1, 0, x], [0, 1, y]])

# Use the OpenCV affine transformation function to implement the
translation operation.
shifted = cv2.warpAffine(img, M, (w, h))

# Return the shifted image.
return shifted

# Load and display the image.
im = cv2.imread('lena.jpg')
cv2.imshow("Orig", im)

# Translate the original image.
# 50 pixels down.
shifted = translate(im, 0, 50)
cv2.imshow("Shift1", shifted)
# 100 pixels left.
shifted = translate(im, -100, 0)
cv2.imshow("Shift2", shifted)
# 50 pixels right and 100 pixels down.
shifted = translate(im, 50, 100)
cv2.imshow("Shift3", shifted)

cv2.waitKey()
cv2.destroyAllWindows()
```

Step 2 Perform rotation.

```
import numpy as np
import cv2

# Define the rotate function.
def rotate(img, angle, center=None, scale=1.0):
    # Obtain the image size.
    (h, w) = img.shape[:2]

    # The missing value of the rotation center is the image center.
    if center is None:
        center = (w / 2, h / 2)

    # Invoke the function of calculating the rotation matrix.
    M = cv2.getRotationMatrix2D(center, angle, scale)

    # Use the OpenCV affine transformation function to implement the rotation
    operation.
    rotated = cv2.warpAffine(img, M, (w, h))

    # Return the rotated image.
    return rotated
```

```
im = cv2.imread('lena.jpg')
cv2.imshow("Orig", im)

# Rotate the original image.
# 45 degrees counterclockwise.
rotated = rotate(im, 45)
cv2.imshow("Rotate1", rotated)
# 20 degrees clockwise.
rotated = rotate(im, -20)
cv2.imshow("Rotate2", rotated)
# 90 degrees counterclockwise.
rotated = rotate(im, 90)
cv2.imshow("Rotate3", rotated)

cv2.waitKey()
cv2.destroyAllWindows()
```

Step 3 Perform mirroring.

```
import numpy as np
import cv2

im = cv2.imread('lena.jpg')
cv2.imshow("orig", im)

# Perform vertical mirroring.
im_flip0 = cv2.flip(im, 0)
cv2.imshow("flip vertical ", im_flip0)

# Perform horizontal mirroring.
im_flip1 = cv2.flip(im, 1)
cv2.imshow("flip horizontal ", im_flip1)

cv2.waitKey()
cv2.destroyAllWindows()
```

Step 4 Perform zooming.

```
import numpy as np
import cv2

im = cv2.imread('lena.jpg')
cv2.imshow("orig", im)

# Obtain the image size.
(h, w) = im.shape[:2]

# Target size.
dst_size = (200, 300)
```

```
# Perform nearest neighbor interpolation.
method = cv2.INTER_NEAREST

# Perform zooming.
resized = cv2.resize(im, dst_size, interpolation = method)
cv2.imshow("resized1", resized)

# Target size.
dst_size = (800,600)

# Perform bilinear interpolation.
method = cv2.INTER_LINEAR

# Perform zooming.
resized = cv2.resize(im, dst_size, interpolation = method)
cv2.imshow("resized2", resized)

cv2.waitKey()
cv2.destroyAllWindows()
```

----End

1.4.4 Grayscale Transformation

Step 1 Perform grayscale transformation, including inversion, grayscale stretch, and grayscale compression.

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

# Define the linear grayscale transformation function.
# k > 1: Stretch the grayscale value.
# 0 < k < 1: Compress the grayscale value.
# k = -1, b = 255: Perform grayscale inversion.
def linear_trans(img, k, b=0):
    # Calculate the mapping table of linear grayscale changes.
    trans_list = [(np.float32(x)*k+b) for x in range(256)]
    # Convert the list to np.array.
    trans_table = np.array(trans_list)
    # Adjust the value out of the range [0,255] and set the data type to
    uint8.
    trans_table[trans_table>255] = 255
    trans_table[trans_table<0] = 0
    trans_table = np.round(trans_table).astype(np.uint8)
    # Use the look up table function in the OpenCV to change the image
    grayscale value.
    return cv2.LUT(img, trans_table)

im = cv2.imread('lena.jpg',0)
cv2.imshow('org', im)
```

```
# Inversion.
im_inversion = linear_trans(im, -1, 255)
cv2.imshow('inversion', im_inversion)
# Grayscale stretch.
im_stretch = linear_trans(im, 1.2)
cv2.imshow('graystretch', im_stretch)
# Grayscale compression.
im_compress = linear_trans(im, 0.8)
cv2.imshow('graycompress', im_compress)
cv2.waitKey()
cv2.destroyAllWindows()
```

Step 2 Perform Gamma transformation.

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

# Define the gamma transformation function.
def gamma_trans(img, gamma):
    # Firstly normalize the input to [0,1], perform the gamma function, and
    then restore the input to [0,255].
    gamma_list = [np.power(x / 255.0, gamma) * 255.0 for x in range(256)]
    # Convert list to np.array and set the data type to uint8.
    gamma_table = np.round(np.array(gamma_list)).astype(np.uint8)
    # Use the look up table function of the OpenCV to change the image
    grayscale value.
    return cv2.LUT(img, gamma_table)

im = cv2.imread('lena.jpg',0)
cv2.imshow('org', im)

# Use the gamma value 0.5 to stretch the shadow and compress the highlight.
im_gama05 = gamma_trans(im, 0.5)
cv2.imshow('gama0.5', im_gama05)
# Use the gamma value 2 to stretch the highlight and compress the shadow.
im_gama2 = gamma_trans(im, 2)
cv2.imshow('gama2', im_gama2)
cv2.waitKey()
cv2.destroyAllWindows()
```

----End

1.4.5 Histogram

Step 1 Display the histogram.

```
import cv2
from matplotlib import pyplot as plt
# Read and display the image.
```

```
im = cv2.imread("lena.jpg",0)
cv2.imshow('org', im)

# Draw a histogram for the grayscale image.
plt.hist(im.ravel(), 256, [0,256])
plt.show()

# Wait for a user to press a key for feedback.
cv2.waitKey()
cv2.destroyAllWindows()
```

Step 2 Perform histogram equalization.

```
import cv2
from matplotlib import pyplot as plt

im = cv2.imread("lena.jpg",0)
cv2.imshow('org', im)

# Invoke the histogram equalization API of the OpenCV.
im_equal = cv2.equalizeHist(im)
cv2.imshow('equal', im_equal)

# Display the histogram of the original image.
plt.subplot(2,1,1)
plt.hist(im.ravel(), 256, [0,256],label='org')
plt.legend()

# Display the histogram of the equalized image.
plt.subplot(2,1,2)
plt.hist(im_equal.ravel(), 256, [0,256],label='equalize')
plt.legend()
plt.show()

# Destroy all windows after the user presses a key for feedback.
cv2.waitKey()
cv2.destroyAllWindows()
```

1.4.6 Filtering

Step 1 Perform median filtering.

```
import cv2
import numpy as np

im = cv2.imread('lena.jpg')
cv2.imshow('org',im)

# Invoke the median filtering API of the OpenCV.
```

```
im_medianblur = cv2.medianBlur(im, 5)

cv2.imshow('median_blur', im_medianblur)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Step 2 Perform mean filtering.

```
# Method 1: Invoke the API of the OpenCV directly.
import cv2
import numpy as np

im = cv2.imread('lena.jpg')
cv2.imshow('org', im)

# Invoke the mean filtering API of the OpenCV.
im_meanblur1 = cv2.blur(im, (3, 3))

cv2.imshow('mean_blur_1', im_meanblur1)

cv2.waitKey(0)
cv2.destroyAllWindows()

# Method 2: Use the mean operator and filter2D to customize filtering
operations.
import cv2
import numpy as np

im = cv2.imread('lena.jpg')
cv2.imshow('org', im)
# Mean operator.
mean_blur = np.ones([3, 3], np.float32)/9

# Use filter2D to perform filtering.
im_meanblur2 = cv2.filter2D(im, -1, mean_blur)
cv2.imshow('mean_blur_2', im_meanblur2)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Step 3 Perform Gaussian filtering.

```
# Method 1: Invoke the API of the OpenCV directly.
import cv2
import numpy as np

im = cv2.imread('lena.jpg')
cv2.imshow('org', im)
```

```
# Invoke the Gaussian filtering API of the OpenCV.
im_gaussianblur1 = cv2.GaussianBlur(im, (5, 5), 0)

cv2.imshow('gaussian_blur_1',im_gaussianblur1)

cv2.waitKey(0)
cv2.destroyAllWindows()

# Method 2: Use the Gaussian operator and filter2D to customize filtering
operations.
import cv2
import numpy as np

im = cv2.imread('lena.jpg')
cv2.imshow('org',im)
# Gaussian operator.
gaussian_blur = np.array([
    [1,4,7,4,1],
    [4,16,26,16,4],
    [7,26,41,26,7],
    [4,16,26,16,4],
    [1,4,7,4,1]], np.float32)/273

# Use filter2D to perform filtering.
im_gaussianblur2 = cv2.filter2D(im,-1,gaussian_blur)
cv2.imshow('gaussian_blur_2',im_gaussianblur2)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Step 4 Perform sharpening.

```
import cv2
import numpy as np

im = cv2.imread('lena.jpg')
cv2.imshow('org',im)
# Sharpening operator 1.
sharpen_1 = np.array([
    [-1,-1,-1],
    [-1,9,-1],
    [-1,-1,-1]])
# Use filter2D to perform filtering.
im_sharpen1 = cv2.filter2D(im,-1,sharpen_1)
cv2.imshow('sharpen_1',im_sharpen1)

# Sharpening operator 2.
sharpen_2 = np.array([
    [0,-1,0],
    [-1,8,-1],
    [0,1,0]])/4.0

# Use filter2D to perform filtering.
im_sharpen2 = cv2.filter2D(im,-1,sharpen_2)
```

```
cv2.imshow('sharpen_2',im_sharpen2)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

----End

1.5 Summary

This section describes how to use the OpenCV image processing library to preprocess images in Python. In this experiment, the OpenCV image processing library is used to implement basic image preprocessing operations, including color space conversion, coordinate transformation, grayscale transformation, histogram transformation, and image filtering. This section can deepen the perception of the image preprocessing technology and provide practical operation guidance for using the technology.

2 Image Recognition

2.1 Experiment Introduction

Image recognition is a technology that uses a computer to process, analyze, and understand images to identify objects in different modes. Image recognition is available through open application programming interfaces (APIs). You can obtain the prediction results by accessing and invoking the APIs in real time. The APIs help you collect key data automatically and build an intelligent service system, thereby improving service efficiency.

Image tagging can recognize more than 3000 types of objects and 20,000 scenarios and tags in natural images, which have extensive semantic meanings because one image contains various tags. It can intelligently and precisely understand image content and make intelligent album management, image retrieval and classification, and scenario- or object-based advertising more accurate.

This service can release the POST RESTful HTTP request service in two modes. One is to invoke the underlying APIs encapsulated by the software development kit (SDK), and the other is to simulate the access to the server from the front-end browser. The first mode needs to use the user's AK/SK for identity authentication, and the second mode needs to obtain the user's token for identity authentication. In this experiment, the AK/SK authentication mode is used to request a service.

2.2 Experiment Objectives

This experiment describes how to use the image recognition function provided by HUAWEI CLOUD services and guides trainees to tag images through image recognition. Currently, HUAWEI CLOUD provides the RESTful API for image recognition and the SDK for image recognition based on Java. Through this experiment, trainees can understand and master how to use Java to develop the image tagging service.

2.3 API Introduction

2.3.1 RESTful API Introduction

HUAWEI CLOUD APIs comply with RESTful API design specifications. Representational State Transfer (REST) allocates Uniform Resource Identifiers (URIs) to dispersed resources so that the resources can be located. Applications on clients use Uniform Resource Locators (URLs) to obtain the resources.

2.3.2 RESTful API Request/Response Composition

A RESTful API request/response consists of the following five parts:

- Request URL

The URL format is as follows: `https://Endpoint/uri`. The parameters in the URL are described in Table 2-1.

Table 2-1 URL parameter description

Parameter	Description
Endpoint	Web service entrance URL. Obtain this value from Regions and Endpoints. Endpoint image.cn-north-1.myhuaweicloud.com corresponding to the image recognition service is used by all service APIs.
uri	Resource path, that is, API access path. Obtain the value from the URI of a specified API, for example, /v1.0/ais/subscribe.

- Request header

The request header consists of two parts: HTTP method and optional additional request header field (such as the field required by a specified URI and HTTP method).

Table 2-2 describes the request methods supported by RESTful APIs.

Table 2-2 Request method description

Method	Description
GET	Requests the server to return specified resources.

PUT	Requests the server to update specified resources.
POST	Requests the server to add resources or perform a special operation.
DELETE	Requests the server to delete specified resources, for example, objects.
PATCH	Requests the server to update partial content of a specified resource. If a target resource does not exist, PATCH may create a resource.

For details about common request headers, see

https://support.huaweicloud.com/en-us/api-asr/asr_03_0013.html.

- Request body

A request body is generally sent in a structured format (for example, JSON or XML), corresponding to Content-type in the request header, and is used to transfer content except the request header. If a request body contains a parameter in Chinese, the parameter must be coded in UTF-8 mode.

- Response header

A response header contains two parts: status code and additional response header field.

Status code, including success codes 2xx and error codes 4xx or 5xx. For details about the status code, see https://support.huaweicloud.com/en-us/api-asr/asr_03_0016.html.

Additional response header field, such as the field required by the response supporting a request (the field in the Content-type response header). For details, see https://support.huaweicloud.com/en-us/api-asr/asr_03_0013.html.

- Response body

A response body is generally returned in a structured format (for example, JSON or XML), and is used to transfer content except the response header. When a service request is successfully sent, the service request result is returned. When a service request fails to be sent, an error code is returned. For details, see https://support.huaweicloud.com/en-us/api-asr/asr_03_0014.html.

2.3.3 Request Initiation Methods

There are three methods to initiate constructed requests, including:

- cURL

cURL is a command line tool, which can be used to perform URL operations and transfer information. cURL functions as an HTTP client can send HTTP requests to the server and receive responses. cURL is applicable to API debugging.

- Code

You can invoke APIs by coding to assemble, send, and process requests.

- Mozilla and Google provide graphical browser plug-ins for REST clients to send and process requests.

2.3.4 Image Tagging API

Function overview

Natural images have rich semantic meanings because one image contains various tags. Image tagging can recognize hundreds of scenarios and thousands of objects and their properties in natural images, making intelligent album management, image retrieval and classification, and scenario- or object-based advertising more intuitive. After the image to be processed is uploaded, image tagging will return the tag and confidence score.

URI

URI format

POST /v1.0/image/tagging

Request

Table 2-3 describes the request parameters.

Table 2-3 Request parameter description

Parameter	Mandatory or Optional	Type	Description
image	Set either this parameter	String	Image data, which is encoded based on Base64. The size of data encoded based on Base64

	or url .		cannot exceed 10 MB. The image resolution of the short edges must be greater than or equal to 15 pixels, and that of the long edges cannot exceed 4096 pixels. The supported image formats include JPG, PNG, and BMP.
url	Set either this parameter or image .	String	URL of the image file. Currently, this URL can be accessed by temporarily authorization on HUAWEI CLOUD OBS or anonymous and public authorization.
language	Optional	String	Language type of the returned tag. The default value is zh , which indicates Chinese.
limit	Optional	Integer	Maximum number of tags that can be returned. The default value is -1 , indicating that all tags are returned.
threshold	Optional	Float	Threshold (0 to 100) of the confidence score. The tags whose confidence score is lower than the threshold will not be returned. The default value is 0 .

Response

Table 2-4 describes the response parameters.

Table 2-4 Response parameter description

Parameter	Type	Description
result	JSON	Content of the image tag returned when the invoking succeeds. The parameter is not included when the API invoking fails.
tags	List	List of tags.
confidence	Float	Confidence score ranging from 0 to 100.
tag	String	Tag name.

error_code	String	Error code returned when the invoking fails. For details, see Error Codes . The parameter is not included when the API invoking succeeds.
error_msg	String	Error message returned when the API invoking fails. The parameter is not included when the API invoking succeeds.

Returned values

- Normal
200
- Failed

Table 2-5 Returned Value parameter description

Returned Value	Description
400	The request cannot be understood by the server due to malformed syntax. A client shall not submit the request again unless the request is modified. The request parameters are incorrect.
401	The request requires user authentication.
403	No permission to perform this operation.
404	The request failed because the requested resource could not be found on the server.
500	The server encountered an unexpected fault which prevented it from processing the request.

2.4 Procedure

In this experiment, you need to download the SDK for image recognition from the HUAWEI CLOUD service platform and use either of the following two methods to

access the SDK. One method is to submit a RESTful service request by invoking the underlying APIs encapsulated by the SDK based on the AK/SK for identity authentication. The other method is to simulate the browser to submit a RESTful request by obtaining the user's token information. The procedure is as follows.

2.4.1 Applying for the Service

- Step 1 **Log in to the HUAWEI CLOUD website and click Console in the upper right corner. The console page is displayed.**
- Step 2 **Click Service List and choose EI Enterprise Intelligence > Image Recognition Service. The Image Recognition Service page is displayed.**
- Step 3 **Click Service Management on the left of the page.**
- Step 4 **Select the Image Tagging service and click Subscribe.**
- Step 5 **After the service is subscribed to, click Image Tagging in the service list on the left to view the service usage procedure.**

----End

2.4.2 Downloading the Image Recognition SDK

Download the SDK software package and documents for image recognition from the following website:

<https://developer.huaweicloud.com/en-us/sdk?IMAGE>

- Step 1 **Log in to the HUAWEI CLOUD website.**

Open a browser, enter the following URL in the address box, and log in to the HUAWEI CLOUD website:

<https://www.huaweicloud.com/en-us/>

- Step 2 **Access the image recognition service of HUAWEI CLOUD.**

Choose Console > Service List > EI Enterprise Intelligence > Image Recognition Service, as shown in the following figure.

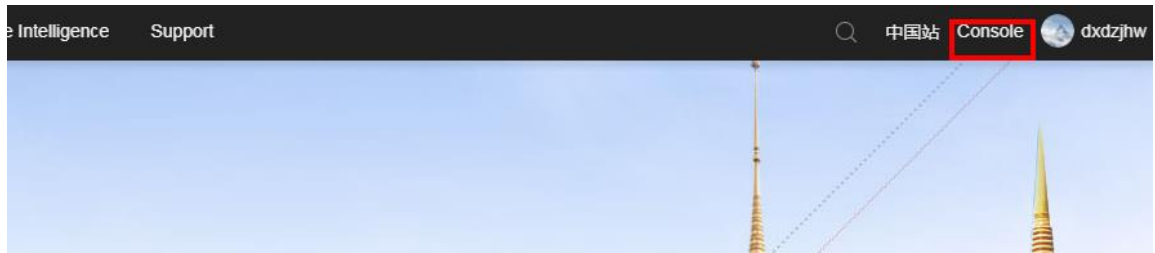


Figure 2-1 Console on the homepage

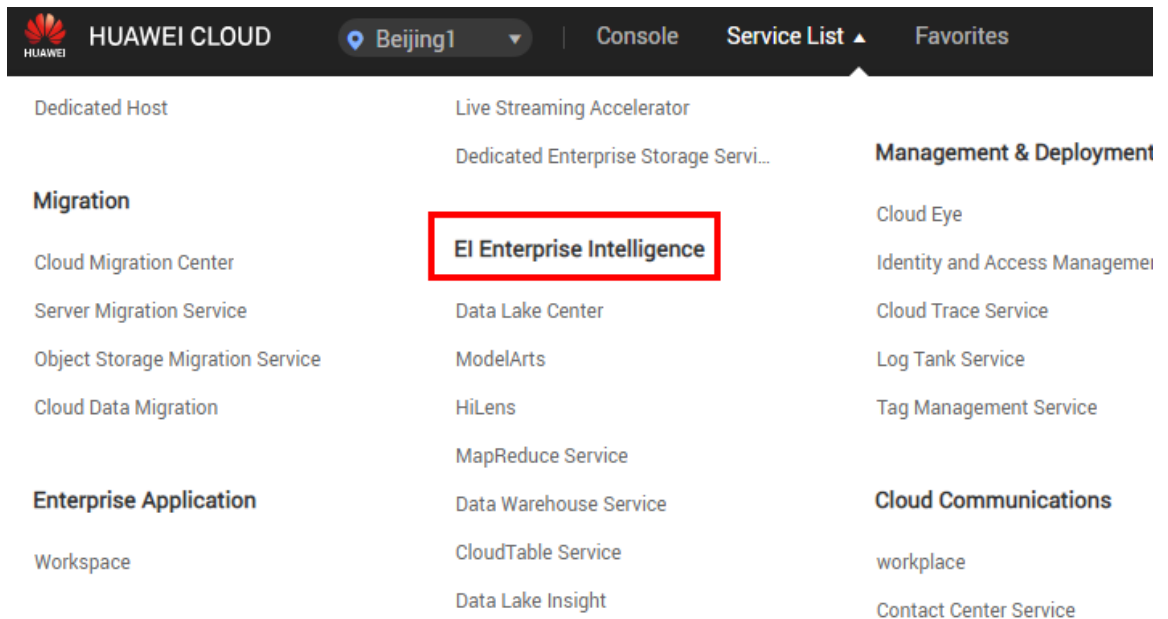


Figure 2-2 EI Enterprise Intelligence

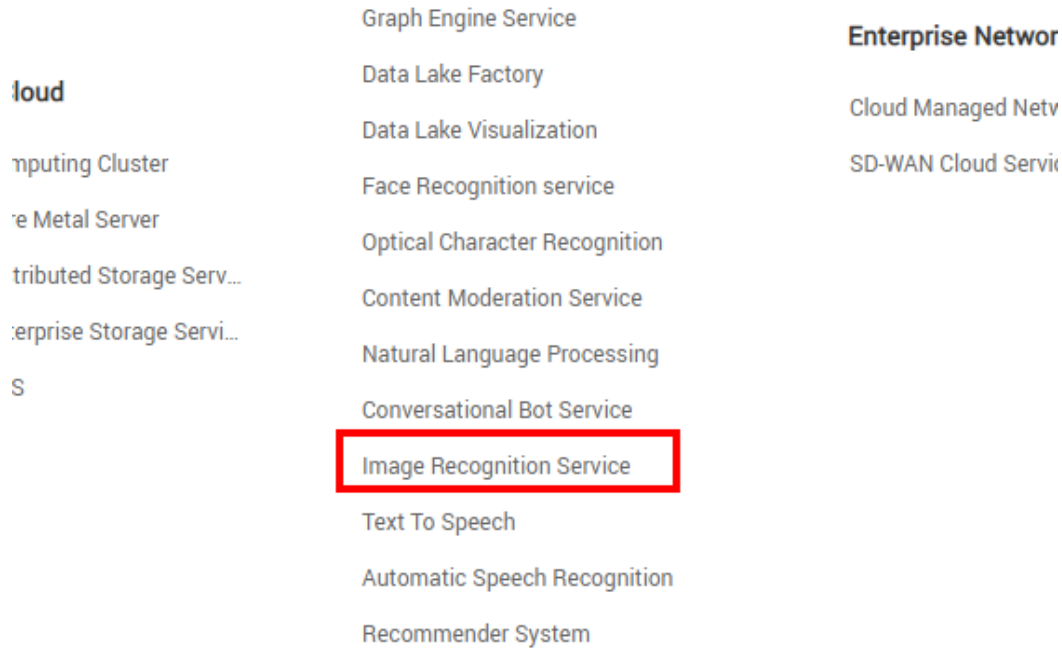


Figure 2-3 Image Recognition Service

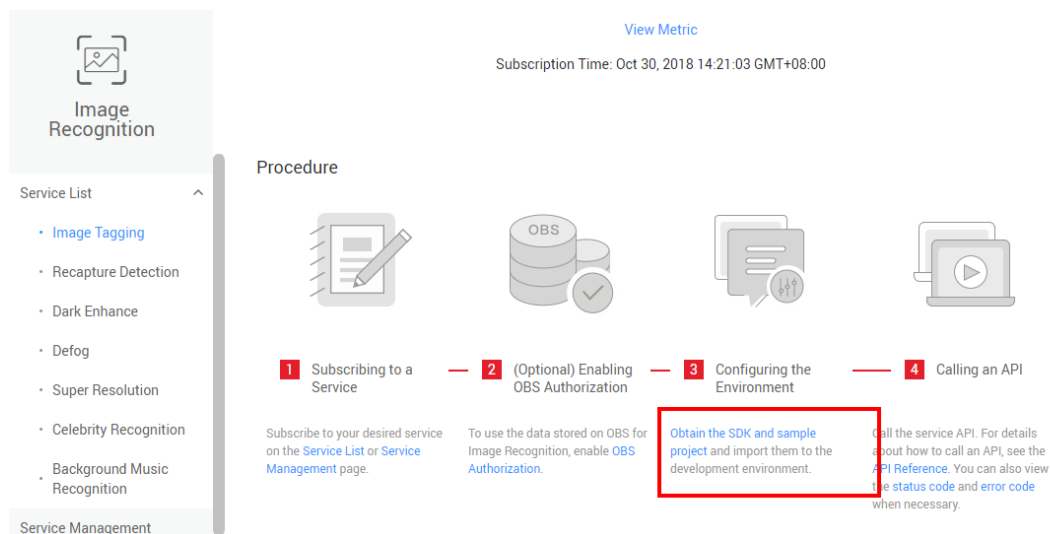


Figure 2-4 Image Recognition Service

Step 3 Search for the image recognition SDK.

Click the SDK link in Figure 2-4 to access the HUAWEI CLOUD SDK. The page shown in Figure 2-5 is displayed.

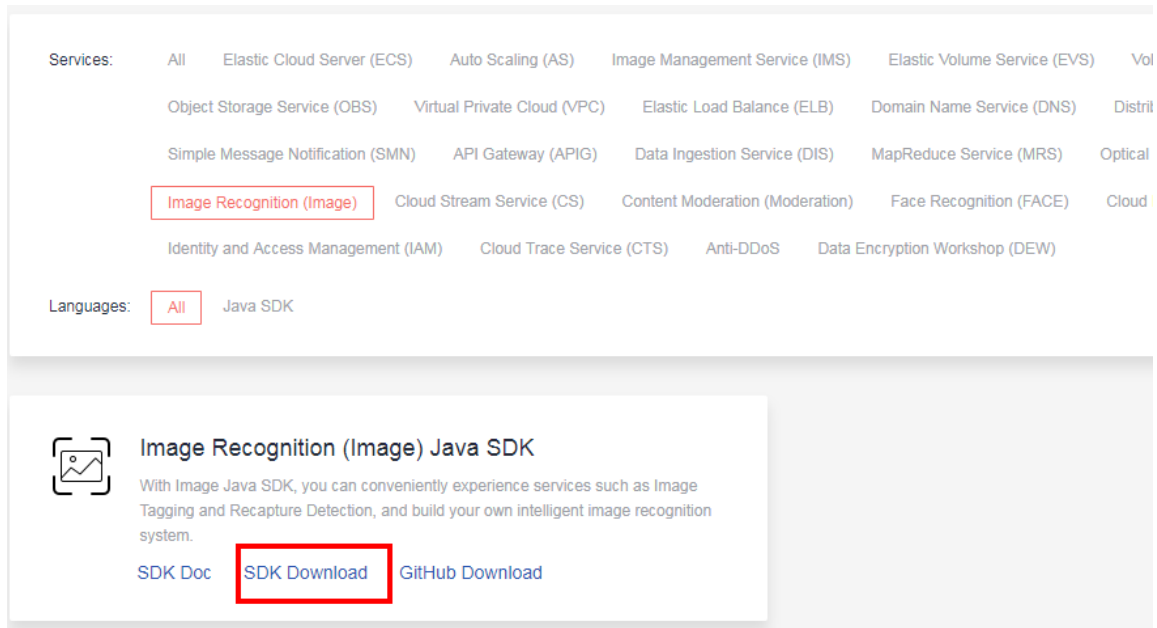


Figure 2-5 HUAWEI CLOUD SDK

Step 4 Download the image recognition SDK.

Click **SDK Download** in the red box in the preceding figure to download the SDK file.

----End

2.4.3 Loading the SDK Project File

Step 1 **Decompress the SDK file and add it to the created workspace.**

Step 2 **Start the Eclipse.**

Step 3 **Ensure that the JRE path configured under Windows > Preferences > Java > Installed JREs is corrected**

Step 4 **Right-click Package Explorer on the left, choose Import from the shortcut menu, choose Maven > Existing Maven Projects, click Next, click Browse, and select the local path where the ais-image-java-sdk is located.**

Step 5 **Click Finish to import the SDK, and open the project after the import succeeds. The project directory is shown in the following figure.**

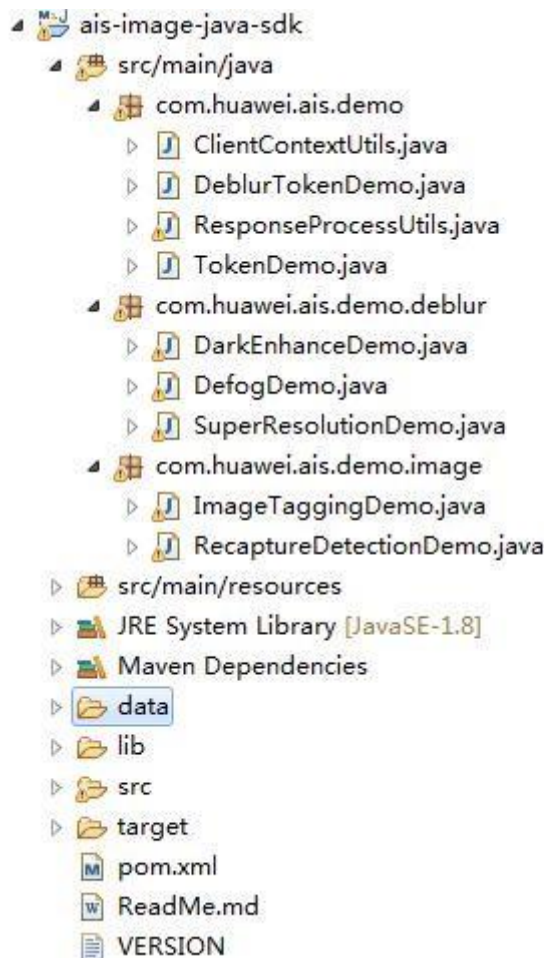


Figure 2-6 The Project Directory

----End

2.4.4 Using the Image Tagging Service Through the AK/SK

Step 1 **Configure the access information for SDK user authentication.**

1. Register an account and log in to the HUAWEI CLOUD console.
2. Move the cursor to the user name and select **My Credential** from the drop-down list box.
3. Click the **Access Keys** tab and click **Add Access Key**.
4. Enter the login password of the current user, use the email or SMS verification mode, and enter the verification code.
5. Click **OK** to download the AK/SK of the authentication account. Keep the AK/SK information secure. You can use the AK/SK information directly for subsequent experiments.

6. Enter the AK/SK of the specified user in **ClientContextUtils.java**, as shown in Figure 2-7.

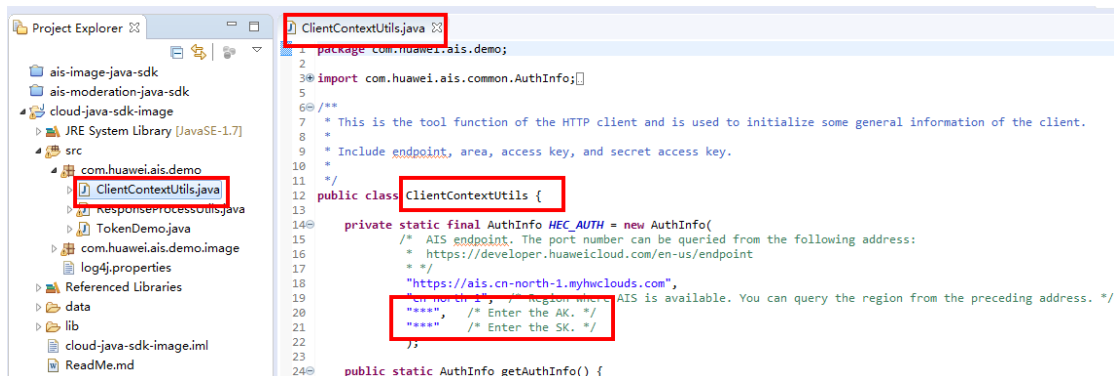


Figure 2-7 AK/SK configuration

- Step 2 **Select a local image or use the default image of the demo. Change the image path in the ImageTaggingDemo.java file by referring to the following figure.**

```
//
// 2. Construct the parameters required for accessing Image Tagging.
//
String uri = "/v1.0/image/tagging";
byte[] fileData = FileUtils.readFileToByteArray(new File("data/image-tagging-demo-1.jpg"));
String fileBase64Str = Base64.encodeBase64String(fileData);
```

Figure 2-8 Image Path Modification Example

- Step 3 **Analyze the ImageTaggingDemo.java class code.**

```
package com.huawei.ais.demo.image;

import com.alibaba.fastjson.JSONObject;
import com.huawei.ais.demo.ClientContextUtils;
import com.huawei.ais.demo.ResponseProcessUtils;
import com.huawei.ais.sdk.AisAccess;
import com.huawei.ais.sdk.AisAccessWithProxy;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.FileUtils;
import org.apache.http.HttpResponse;
import org.apache.http.entity.StringEntity;

import java.io.File;
import java.io.IOException;

/**
 * Example class of using the image tagging service.
```

```
*/
public class ImageTaggingDemo {
//
// Example function of using the image tagging service.
//
private static void imageTaggingDemo() throws IOException {
//
// 1. In the ClientContextUtils class, configure basic information for
accessing the image tagging service.
// Generate a client connection object.
//
// Configure the following three timeout limits:
int connectionTimeout = 5000; // Timeout limit for connecting to the
target URL.
int connectionRequestTimeout = 1000; // Timeout limit for obtaining
available connections from the connection pool.
int socketTimeout = 5000; // Timeout limit for obtaining a server
response.

AisAccess service = new AisAccess(ClientContextUtils.getAuthInfo(),
connectionTimeout, connectionRequestTimeout, socketTimeout);

//
// 1.a The image tagging service can be accessed in proxy mode. It
applies to the scenario where a user cannot directly access HUAWEI CLOUD
services, for example, when the intranet is used.
// If a proxy is used, you need to configure the parameter class
(ProxyHostInfo) of the proxy server in ClientContextUtils.
//
//AisAccess service = new
AisAccessWithProxy(ClientContextUtils.getAuthInfo(),
ClientContextUtils.getProxyHost(),
connectionTimeout, connectionRequestTimeout, socketTimeout);

try {
//
// 2. Construct the parameters required for accessing the image
tagging service.
//
String uri = "/v1.0/image/tagging";
byte[] fileData = FileUtils.readFileToByteArray(new File("data/image-
tagging-demo-1.jpg"));
//byte[] fileData = FileUtils.readFileToByteArray(new
File("data/1356924004_7530.jpg"));

String fileBase64Str = Base64.encodeBase64String(fileData);

JSONObject json = new JSONObject();
json.put("image", fileBase64Str);
json.put("threshold", 60);
StringEntity stringEntity = new StringEntity(json.toJSONString(),
"utf-8");

// 3. Pass the URI parameter and required parameters of the image
tagging service.
```

```
// Pass the parameters in JSON objects and invoke the service using
POST.

HttpResponse response = service.post(uri, stringEntity);

// 4. Check whether the service is invoked successfully. If 200 is
returned, the invoking succeeds. Otherwise, it fails.
ResponseProcessUtils.processResponseStatus(response);

// 5. Process the character stream returned by the service and
generate the recognition result.
ResponseProcessUtils.processResponse(response);
} catch (Exception e) {
    e.printStackTrace();
} finally {

// 6. Disconnect the client.
service.close();
}

}

//
// Main entry point function.
//
public static void main(String[] args) throws IOException {
    // Test entry point function.
    imageTaggingDemo();
}
}
```

Step 4 Run the example program.

Right-click **ImageTaggingDemo.java** in **Project Explorer**, choose **Run** from the menu bar, and click **Run As** to run the program. If **200** is displayed on the console, the program is successfully executed. The image tag recognition result is delivered to the console.

```
C:\Java\jdk1.8.0_112\bin\java ...
200
{"result": {"tags": [{"confidence": 82.9215240478516, "tag": "beach"}, {"confidence": 62.9527206420898, "tag": "sand"}, {"confidence": 61.7384376525879, "tag": "ocean"}, {"confidence": 61.1379547119141, "tag": "sea"}]}}
```

Figure 2-9 View Program running result

Step 5 the running result.

1. Response success example

```
{
  "result": {
    "tags": [
      {
        "confidence": 38.51,
```

```
        "tag": "sky"
    },
    {
        "confidence": 25.75,
        "tag": "landscape"
    }
]
}
```

2. Response failure example

```
{
    "error_code": "AIS.0005",
    "error_msg": "The service does not exist."
}
----End
```

2.5 Summary

This section describes how to use the HUAWEI CLOUD image tagging service for experiment operations. The SDK is used to send RESTful requests to implement related functions. When using the SDK to initiate a RESTful request, configure the user authentication information. Two authentication modes are available: AK/SK and token. This section describes the AK/SK systematically, providing an operation guide for using the image tagging service.

3

Content Moderation

3.1 Experiment Introduction

With the rapid development of the Internet and the increase of information, a large amount of bad information such as pornography, violence, and political sensitivity is mixed. If the content is not reviewed, users may be unsatisfied with the content and gradually reduce the frequency of using or even give up the products. Content moderation is to review the images, texts, and videos uploaded by users, and automatically identifies pornographic, violent, and politically sensitive information. Content moderation is available through open APIs. You can obtain the prediction results by accessing and invoking the APIs in real time. The APIs help you collect key data automatically and build an intelligent service system, thereby improving service efficiency.

Image content moderation uses the Deep Neural Network models to predict the image content that violates the regulations and accurately identify the political figures, violent and terrorist elements, and pornographic content in the images, eliminating risks.

This service can release the POST RESTful HTTP request service in two modes. One is to invoke the underlying APIs encapsulated by the SDK, and the other is to simulate the access to the server from the front-end browser. The first mode needs to use the user's AK/SK for identity authentication, and the second mode needs to obtain the user's token for identity authentication. In this experiment, the AK/SK authentication mode is used to request a service.

3.2 Experiment Objectives

This experiment describes how to use the content moderation function provided by HUAWEI CLOUD services and guides trainees to detect image content through the content moderation function. Currently, HUAWEI CLOUD provides the RESTful API for image content moderation and the SDK for image content moderation based on Java.

Through this experiment, trainees can understand and master how to use Java to develop the image content moderation service.

3.3 API Introduction

3.3.1 RESTful API Introduction

HUAWEI CLOUD APIs comply with RESTful API design specifications. Representational State Transfer (REST) allocates Uniform Resource Identifiers (URIs) to dispersed resources so that the resources can be located. Applications on clients use Uniform Resource Locators (URLs) to obtain the resources.

3.3.2 RESTful API Request/Response Composition

A RESTful API request/response consists of the following five parts:

- Request URL

The URL format is as follows: `https://Endpoint/uri`. The parameters in the URL are described in Table 3-1.

Table 3-1 URL parameter description

Parameter	Description
Endpoint	Web service entrance URL. Obtain this value from Regions and Endpoints . Endpoint moderation.cn-north-1.myhuaweicloud.com corresponding to content moderation is used by all service APIs.
uri	Resource path, that is, API access path. Obtain the value from the URI of a specified API, for example, /v1.0/ais/subscribe.

- Request header

The request header consists of two parts: HTTP method and optional additional request header field (such as the field required by a specified URI and HTTP method).

Table 3-2 describes the request methods supported by RESTful APIs.

Table 3-2 Request method description

Method	Description
GET	Requests the server to return specified resources.
PUT	Requests the server to update specified resources.
POST	Requests the server to add resources or perform a special operation.
DELETE	Requests the server to delete specified resources, for example, objects.
PATCH	Requests the server to update partial content of a specified resource. If a target resource does not exist, PATCH may create a resource.

For details about common request headers, see https://support.huaweicloud.com/en-us/api-asr/asr_03_0013.html.

- Request body

A request body is generally sent in a structured format (for example, JSON or XML), corresponding to Content-type in the request header, and is used to transfer content except the request header. If a request body contains a parameter in Chinese, the parameter must be coded in UTF-8 mode.

- Response header

A response header contains two parts: status code and additional response header field.

Status code, including success codes 2xx and error codes 4xx or 5xx. For details about the status code, see https://support.huaweicloud.com/en-us/api-asr/asr_03_0016.html. Additional response header field, such as the field required by the response supporting a request (the field in the Content-type response header). For details, see https://support.huaweicloud.com/en-us/api-asr/asr_03_0013.html.

- Response body

A response body is generally returned in a structured format (for example, JSON or XML), and is used to transfer content except the response header. When a service request is successfully sent, the service request result is returned. When a service

request fails to be sent, an error code is returned. For details, see https://support.huaweicloud.com/en-us/api-asr/asr_03_0014.html.

3.3.3 Request Initiation Methods

There are three methods to initiate constructed requests, including:

- cURL

cURL is a command line tool, which can be used to perform URL operations and transfer information. cURL functions as an HTTP client to send HTTP requests to the server and receive responses. cURL is applicable to API debugging. For details about cURL, see <https://curl.haxx.se/>.

- Code

You can invoke APIs by coding to assemble, send, and process requests.

- Mozilla and Google provide graphical browser plug-ins for REST clients to send and process requests.

For Firefox, see [Firefox RESTClient](#).

For Chrome, see Postman Interceptor.

3.3.4 Image Content Moderation API

Function overview

This API analyzes and identifies whether sensitive content (such as political figures, violent and terrorist elements, and pornographic content) exists in images uploaded by users, and returns identification results.

URI

URI format

POST /v1.0/moderation/image

Request

Table 3-3 describes the request parameters.

Table 3-3 Request parameter description

Parameter	Mandatory or Optional	Type	Description
image	Set either this	String	Image data, which is encoded based

	parameter or url .		on Base64. The size of data encoded based on Base64 cannot exceed 10 MB. The face of a political figure must be greater than or equal to 50 x 50 pixels. The supported image formats include JPG, PNG, and BMP.
url	Set either this parameter or image .	String	URL of the image file. Currently, this URL can be accessed by temporarily authorization on HUAWEI CLOUD OBS or anonymous and public authorization.
categories	Optional	Array	Detection scenario. The options are as follows: • politics: checks whether political figures exist. • terrorism: checks whether terrorism-related content exists. • porn: checks whether pornographic elements exist. • all: checks all elements including the politics, terrorism, and porn scenarios. You can set the parameter to the one of preceding values to check the corresponding elements. If this parameter is left blank or does not exist, both politics and terrorism scenarios are checked, but the porn scenario is not included. NOTE The number of detection times in each scenario is calculated by category.

threshold	Optional	Number	Threshold for filtering the detection results. Only the results whose confidence score is greater than or equal to the threshold are displayed in the detail list. If this parameter is not specified, the default value of each scenario is used. The default value of the politics scenario is 0.48, and that of the terrorism scenario is 0. If there is no special requirement, leave the parameter blank, as used in the sample request, or do not pass the parameter. NOTE • If the maximum confidence score of a specific detection scenario is smaller than the value of threshold, the detail list is empty. If the value of threshold is too small, too many results will be displayed in the detail list. • The threshold parameter does not affect suggestion in the response.
-----------	----------	--------	--

Response

Table 3-4 describes the response parameters.

Table 3-4 Response parameter description

Parameter	Type	Description
result	Object	Invoking result when the invoking succeeds. The parameter is not included when the API invoking fails.
detail	Object	List of detection results of a specific scenario, where politics lists the detection results of political figures, terrorism lists the detection results of terrorism-

		related content, and porn lists the detection results of pornographic elements. If the maximum confidence score in a detection scenario does not reach the threshold, the result list is empty.
confidence	Number	Confidence score. The value range is 0 to 1.
face_detail	Object	Facial information in the politics scenario, including the following four attributes: • h: height of the face area. • w: width of the face area. • x: distance from the upper left corner of the face area to the y axis. • y: distance from the upper left corner of the face area to the x axis.
label	String	Label of each detection result. • In the politics scenario, label indicates the information about the corresponding political figure. • In the terrorism scenario, label indicates the terrorism-related content (such as guns, knives, or fire). • In the porn scenario, label indicates pornographic elements (such as porn information or sexy information). • In the ad scenario, label indicates the advertisement detection result (common content or advertisement).
suggestion	String	Indicates whether an image passes the check. The options are as follows: • block: Sensitive information is detected and therefore the image fails to pass the check. • pass: No sensitive information is detected and therefore the image passes the check.

		• review: Manual review is required. NOTE When multiple scenarios are detected at the same time, the value of suggestion is subject to the scenario where sensitive information is most likely to be included. That is, if a block occurs in any scenario, the value of suggestion is block. If all scenarios pass the check, the value of suggestion is pass. In addition, if manual review is required in any scenario, the value of suggestion is review.
error_code	String	Error code returned when the invoking fails. For details, see Error Codes. The parameter is not included when the API invoking succeeds.
error_msg	String	Error message returned when the API invoking fails. The parameter is not included when the API invoking succeeds.

Returned values

- Normal
200
- Failed

Table 3-5 Returned Value parameter description

Returned Value	Description
400	The request cannot be understood by the server due to malformed syntax. A client shall not submit the request again unless the request is modified. The request parameters are incorrect.
401	The request requires user authentication.
403	No permission to perform this operation.

404	The request failed because the requested resource could not be found on the server.
500	The server encountered an unexpected fault which prevented it from processing the request.

3.4 Procedure

In this experiment, you need to download the SDK for content moderation from the HUAWEI CLOUD service platform and use either of the following two methods to access the SDK. One method is to submit a RESTful service request by invoking the underlying APIs encapsulated by the SDK based on the AK/SK for identity authentication. The other method is to simulate the browser to submit a RESTful request by obtaining the user's token information. The procedure is as follows.

3.4.1 Applying for the Service

- Step 1 **Log in to the HUAWEI CLOUD website and click Console in the upper right corner. The console page is displayed.**
- Step 2 **Click Service List and choose EI Enterprise Intelligence > Content Moderation Service > Image Moderation. The Image Moderation page is displayed.**
- Step 3 **Click Subscribe on the page.**
- End

3.4.2 Downloading the Image Content Moderation SDK

Download the SDK software package and documents for image content moderation from the following website:

<https://developer.huaweicloud.com/en-us/sdk?MODERATION>

- Step 1 **Log in to the HUAWEI CLOUD website.**

Open a browser, enter the following URL in the address box, and log in to the HUAWEI CLOUD website:

<https://www.huaweicloud.com/en-us/>

Step 2 Access the content moderation service of HUAWEI CLOUD.

Choose Console > Service List > EI Enterprise Intelligence > Content Moderation Service, as shown in the following figure.

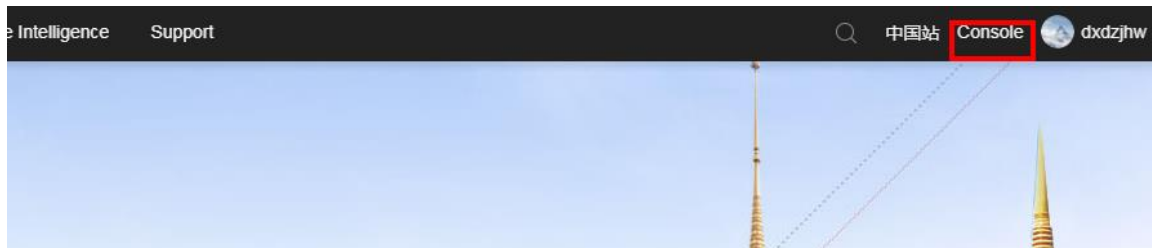


Figure 3-1 Console on the homepage

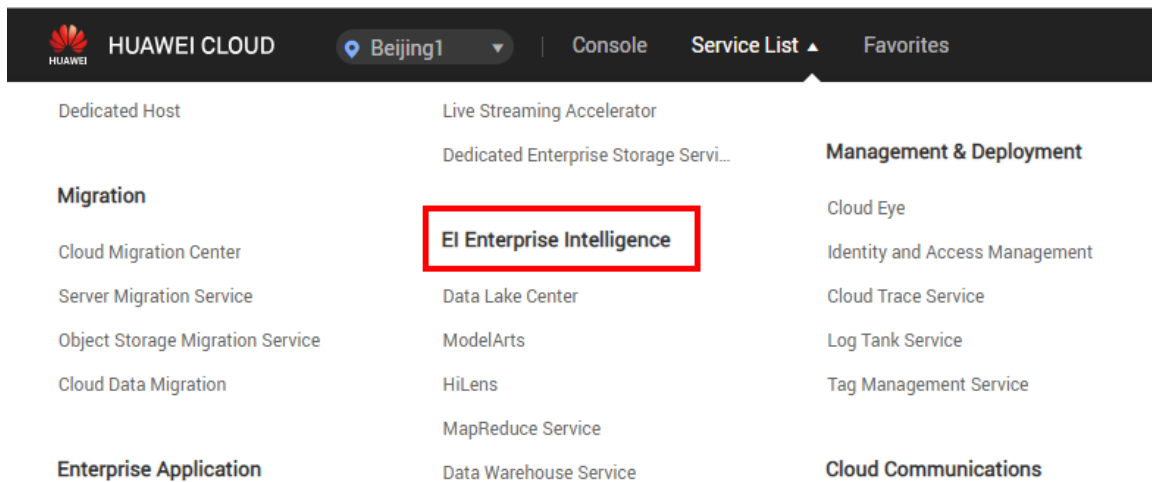
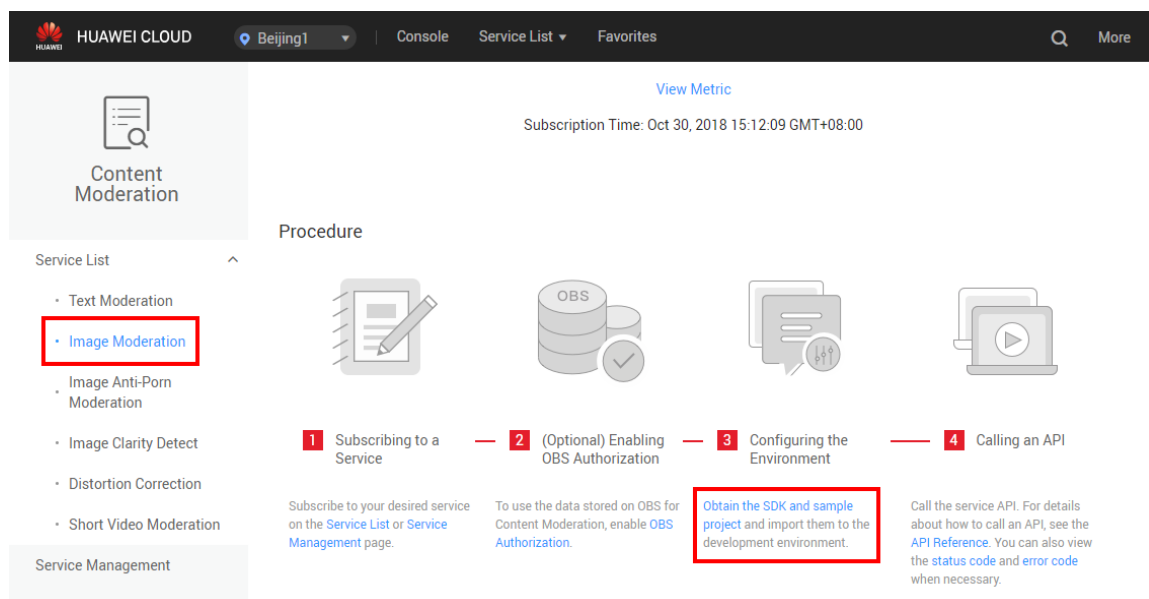


Figure 3-2 EI Enterprise Intelligence

Video Media Processing Center Intelligent Video Surveillance Video on Demand Live	Cloud Stream Service Data Ingestion Service Deep Learning Service Cloud Search Service Machine Learning Service Graph Engine Service Data Lake Factory Data Lake Visualization Face Recognition service Optical Character Recognition Content Moderation Service Natural Language Processing Conversational Bot Service Image Recognition Service	IoT IoT Platform Intelligent EdgeFabric
Dedicated Cloud Dedicated Computing Cluster Dedicated Bare Metal Server Dedicated Distributed Storage Serv... Dedicated Enterprise Storage Servi... Dedicated OBS		Enterprise Network Cloud Managed Netwo SD-WAN Cloud Service

Figure 3-3 Content Moderation Service



HUAWEI CLOUD | Beijing1 | Console | Service List | Favorites | Search | More

Content Moderation

Subscription Time: Oct 30, 2018 15:12:09 GMT+08:00

Procedure

- 1 Subscribing to a Service
- 2 (Optional) Enabling OBS Authorization
- 3 Configuring the Environment
- 4 Calling an API

Service List

- Text Moderation
- Image Moderation**
- Image Anti-Porn Moderation
- Image Clarity Detect
- Distortion Correction
- Short Video Moderation

Service Management

Subscribe to your desired service on the [Service List](#) or [Service Management](#) page.

To use the data stored on OBS for Content Moderation, enable [OBS Authorization](#).

Obtain the SDK and sample project and import them to the development environment.

Call the service API. For details about how to call an API, see the [API Reference](#). You can also view the [status code](#) and [error code](#) when necessary.

Figure 3-4 Content Moderation Service

Step 3 Search for the content moderation service SDK.

Click the SDK link in Figure 3-4 to access the HUAWEI CLOUD SDK. The page shown in Figure 3-5 is displayed.

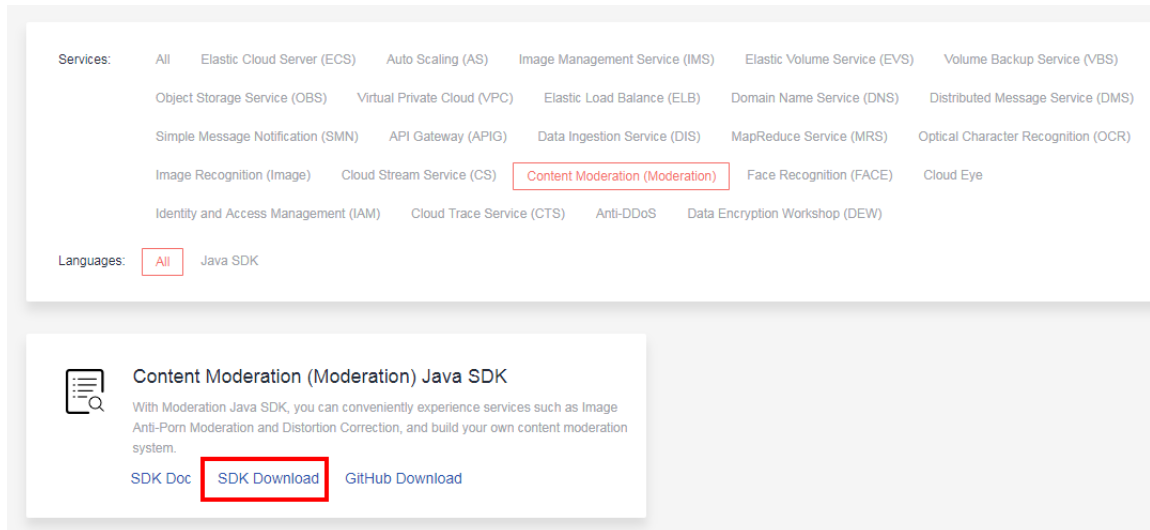


Figure 3-5 HUAWEI CLOUD SDK

Step 4 Download the content moderation service SDK.

Click **SDK Download** in the red box in the preceding figure to download the SDK file.

-----End

3.4.3 Loading the SDK Project File

Step 1 Decompress the SDK file and add it to the created workspace.

Step 2 Start the Eclipse.

Step 3 Ensure that the correct JRE path is configured under Windows > Preferences > Java > Installed JREs.

Step 4 Right-click Package Explorer on the left, choose Import from the shortcut menu, choose Maven > Existing Maven Projects, click Next, click Browse, and select the local path where the ais-moderation-java-sdk is located.

Step 5 Click Finish to import the SDK, and open the project after the import succeeds. The project directory is shown in the following figure.

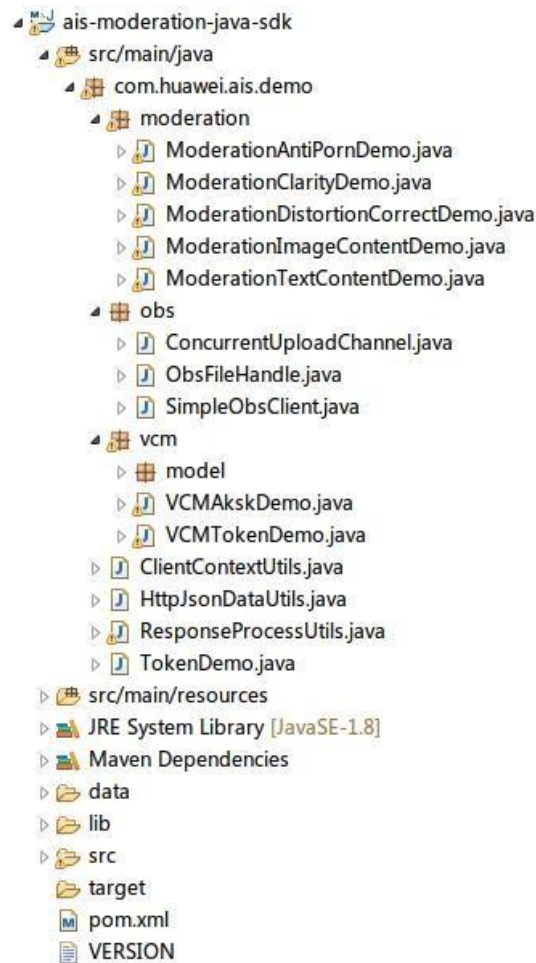


Figure 3-6 The Project Directory

----End

3.4.4 Using the Image Content Moderation Service Through the AK/SK

Step 1 Configure the access information for SDK user authentication.

1. Register an account and log in to the HUAWEI CLOUD console.
2. Move the cursor to the user name and select **My Credential** from the drop-down list box.
3. Click the **Access Keys** tab and click **Add Access Key**.
4. Enter the login password of the current user, use the email or SMS verification mode, and enter the verification code.

5. Click **OK** to download the AK/SK of the authentication account. Keep the AK/SK information secure. You can use the AK/SK information directly for subsequent experiments.
6. Enter the AK/SK of the specified user in **ClientContextUtils.java**, as shown in Figure 3-7.

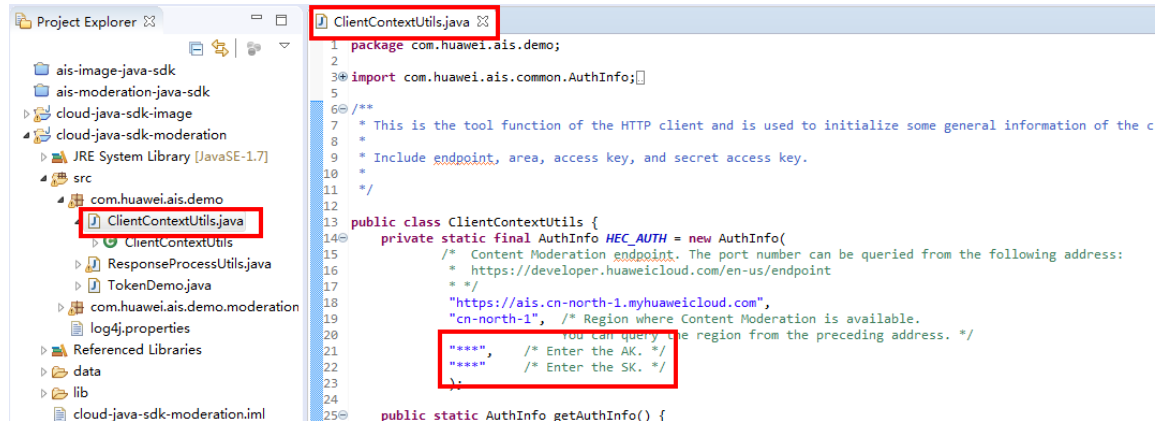


Figure 3-7 AK/SK configuration

Step 2 Select a local image or use the default image of the demo. Change the image path in the ModerationImageContentDemo.java file by referring to the following figure.

```
public static void main(String[] args) throws IOException {
    ModerationImageContentDemo tool = new ModerationImageContentDemo();
    byte[] imageBytes = tool.downloadUrl("http://moderation-demo.ei.huaweicloud.com/theme/images/imagga/image_tagging_04.jpg");
    tool.imageContentCheck(imageBytes);
    tool.imageAntiporn(imageBytes);

    imageBytes = FileUtils.readFileToByteArray(new File("data/moderation-demo-1.jpg"));
    tool.imageContentCheck(imageBytes);
    tool.imageAntiporn(imageBytes);
}
```

Figure 3-8 Image Path Modification Example

Step 3 Analyze the ModerationImageContentDemo.java class code.

```
package com.huawei.ais.demo.moderation;

import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
import com.huawei.ais.demo.ClientContextUtils;
import com.huawei.ais.demo.ResponseProcessUtils;
import com.huawei.ais.sdk.AisAccess;
import com.huawei.ais.sdk.AisAccessWithProxy;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.FileUtils;
```

```
import org.apache.http.HttpResponse;
import org.apache.http.entity.StringEntity;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;

/**
 * Example class of using the image moderation service.
 */
public class ModerationImageContentDemo {

    //
    // 1. In the ClientContextUtils class, configure basic information for
    // accessing the image moderation service.
    // Generate a client connection object.
    // Configure the following three timeout limits:
    private int connectionTimeout = 5000; // Timeout limit for connecting to the
    target URL.
    private int connectionRequestTimeout = 1000; // Timeout limit for obtaining
    available connections from the connection pool.
    private int socketTimeout = 5000; // Timeout limit for obtaining a server
    response.

    private AisAccess service;

    public ModerationImageContentDemo() {
        service = new AisAccess(ClientContextUtils.getAuthInfo(),
        connectionTimeout, connectionRequestTimeout, socketTimeout);
        //
        // 1.a The image moderation service can be accessed in proxy mode. It
        // applies to the scenario where a user cannot directly access HUAWEI CLOUD
        // services, for example, when the intranet is used.
        // If a proxy is used, you need to configure the parameter class
        // (ProxyHostInfo) of the proxy server in ClientContextUtils.
        //
        // service = new AisAccessWithProxy(ClientContextUtils.getAuthInfo(),
        // ClientContextUtils.getProxyHost(),
        // connectionTimeout, connectionRequestTimeout, socketTimeout);
    }

    private void imageContentCheck(byte[] imagebytes) throws IOException {
        try {
            //
            // 2. Initialize the parameters required for accessing the image
            // moderation service.
            //
            String uri = "/v1.0/moderation/image";

            String fileBase64Str = Base64.encodeBase64String(imagebytes);
```

```
JSONObject json = new JSONObject();

json.put("image", fileBase64Str);

StringEntity stringEntity = new StringEntity(json.toJSONString(),
"utf-8");

// 3. Pass the URI parameter and required parameters of the image
moderation service.
// Pass the parameters in JSON objects and invoke the service using
POST.
HttpResponse response = service.post(uri, stringEntity);

// 4. Check whether the service is invoked successfully. If 200 is
returned, the invoking succeeds. Otherwise, it fails.
ResponseProcessUtils.processResponseStatus(response);

// 5. Process the character stream returned by the service and
generate the recognition result.
ResponseProcessUtils.processResponse(response);
} catch (Exception e) {
    e.printStackTrace();
} finally {

    // 6. Disconnect the client.
    service.close();
}
}

public byte[] downloadUrl(String url) throws MalformedURLException,
IOException {
    InputStream in = new URL(url).openStream();
    ByteArrayOutputStream out = new ByteArrayOutputStream();

    byte[] buffer = new byte[128];
    int n = in.read(buffer);
    while (n != -1) {
        out.write(buffer, 0, n);
        n = in.read(buffer);
    }
    in.close();
    out.close();

    return out.toByteArray();
}

//
// Main entry point function.
//
public static void main(String[] args) throws IOException {
    ModerationImageContentDemo tool = new ModerationImageContentDemo();
    byte[] imageBytes = tool.downloadUrl("http://moderation-
demo.ei.huaweicloud.com/theme/images/imagga/image_tagging_04.jpg");
}
```

```

tool.imageContentCheck(imageBytes);

imageBytes = FileUtils.readFileToByteArray(new File("data/moderation-
demo-1.jpg"));
tool.imageContentCheck(imageBytes);

}
}

```

Based on the API request parameter code, the review results of different functions can be displayed.

```

String uri = "/v1.0/moderation/image";

String fileBase64Str = Base64.encodeBase64String(imagebytes);

JSONObject json = new JSONObject();
String [] strArray = new String [1];
strArray[0]="all";

json.put("image", fileBase64Str);
json.put("categories", strArray);

StringEntity stringEntity = new StringEntity(json.toJSONString(), "utf-8");

```

Figure 3-9 API request parameter code

Step 4 Run the example program.

Right-click **ModerationImageContentDemo.java** in **Project Explorer**, choose **Run** from the menu bar, and click **Run As** to run the program. If **200** is displayed on the console, the program is successfully executed. The result is delivered to the console.

```

200
{"result":{"detail":{"politics":[],"terrorism":[{"confidence":0.0,"label":"bloody"}, {"confidence":0.0,"label":"fire"}, {"
confidence":0.0,"label":"gun"}, {"confidence":0.0,"label":"knife"}, {"confidence":0.0,"label":"flag"}, {"confidence":0.0,"l
abel":"symbol"}, {"confidence":0.0,"label":"dress"}, {"confidence":0.0,"label":"war"}, {"confidence":0.0,"label":"tiananmen
"}, {"confidence":1.0,"label":"normal"}]}, "suggestion":"pass"}}

```

Figure 3-10 Program running result

Step 5 View the running result.

1. Response success example

```

{
  "result": {
    "detail": {
      "politics": [

```

```
{
  "confidence": 0.89155281782150264,
  "face_detail":{
    "h": 130,
    "w": 103,
    "x": 80,
    "y": 125
  },
  "label": "xxx"
},
{
  "confidence": 0.87563462432654422,
  "face_detail":{
    "h": 131,
    "w": 203,
    "x": 423,
    "y": 109
  },
  "label": "yyy"
}
],
"terrorism":[
  {
    "confidence": 0.73155281782150264,
    "label": "xxx"
  },
  {
    "confidence": 0.26844718217849736,
    "label": "yyy"
  }
]
},
"suggestion": "block"
}
```

2. Response failure example

```
{
```

```
"error_code": "AIS.0005",  
"error_msg": "The service does not exist."  
}  
----End
```

3.5 Summary

This section describes how to use the HUAWEI CLOUD image moderation service for experiment operations. The SDK is used to send RESTful requests to implement related functions. When using the SDK to initiate a RESTful request, configure the user authentication information. Two authentication modes are available: AK/SK and token. This section describes the AK/SK systematically, providing an operation guide for using the image moderation service.

4 OCR

4.1 Experiment Introduction

Optical Character Recognition (OCR) allows you to detect and recognize printed characters in images and convert such characters into editable text. OCR is available through open APIs. You can obtain the prediction results by accessing and invoking the APIs in real time. The APIs help you collect key data automatically and build an intelligent service system, thereby improving service efficiency.

General text OCR extracts text and its location from an image, and converts the text into structured data according to the location information.

ID card OCR automatically recognizes all information on both sides of ID cards. After an ID card is scanned, information including the ID number, name, and address can be identified even under complex conditions such as low light, tilt, overexposure, and shadow.

VAT invoice OCR recognizes text on value-added tax (VAT) invoices for further processing by using technologies, such as image preprocessing, table extraction, text extraction, text recognition, and structured information output, greatly saving labor costs.

This service can release the POST RESTful HTTP request service in two modes. One is to invoke the underlying APIs encapsulated by the SDK, and the other is to simulate the access to the server from the front-end browser. The first mode needs to use the user's AK/SK for identity authentication, and the second mode needs to obtain the user's token for identity authentication. In this experiment, the AK/SK authentication mode is used to request a service.

4.2 Experiment Objectives

This experiment describes how to use the OCR function provided by HUAWEI CLOUD services. Trainees will learn how to use the OCR service to identify general texts, ID

cards, and value-added tax invoices. Currently, HUAWEI CLOUD provides the RESTful API for OCR and the SDK for content moderation based on Java. Through this experiment, trainees can understand and master how to use Java to develop the OCR service.

4.3 API Introduction

4.3.1 RESTful API Introduction

HUAWEI CLOUD APIs comply with RESTful API design specifications. Representational State Transfer (REST) allocates Uniform Resource Identifiers (URIs) to dispersed resources so that the resources can be located. Applications on clients use Uniform Resource Locators (URLs) to obtain the resources.

4.3.2 RESTful API Request/Response Composition

A RESTful API request/response consists of the following five parts:

- Request URL

The URL format is as follows: `https://Endpoint/uri`. The parameters in the URL are described in Table 4-1.

Table 4-1 URL parameter description

Parameter	Description
Endpoint	Web service entrance URL. Obtain this value from Regions and Endpoints . Endpoint ocr.cn-north-1.myhuaweicloud.com corresponding to OCR is used by all service APIs.
uri	Resource path, that is, API access path. Obtain the value from the URI of a specified API, for example, <code>/v1.0/ais/subscribe</code> .

- Request header

The request header consists of two parts: HTTP method and optional additional request header field (such as the field required by a specified URI and HTTP method).

Table 4-2 describes the request methods supported by RESTful APIs.

Table 4-2 Request method description

Method	Description
GET	Requests the server to return specified resources.
PUT	Requests the server to update specified resources.
POST	Requests the server to add resources or perform a special operation.
DELETE	Requests the server to delete specified resources, for example, objects.
PATCH	Requests the server to update partial content of a specified resource. If a target resource does not exist, PATCH may create a resource.

4.3.3 General Text OCR API

Function overview

This API recognizes the text on an image that has been uploaded (or the URL redirecting to the image file stored on OBS), and returns the result to the user.

URI

URI format

POST /v1.0/ocr/general-text

Request

Table 4-3 describes the request parameters.

Table 4-3 Request parameter description

Parameter	Mandatory or Optional	Type	Description
image	Set either this parameter or url .	String	Image data, which is encoded based on Base64. The size of data encoded based on Base64 cannot exceed 10 MB. The pixels of each side of an image range

			from 15 to 4096. The supported image formats include JPG, PNG, BMP, and TIFF.
url	Set either this parameter or image .	String	URL of the image file. Currently, this URL can be accessed by temporarily authorization on HUAWEI CLOUD OBS or anonymous and public authorization.
detect_direction	Optional	Boolean	Switch for detecting the image direction. The options are as follows: • true: Detect the image direction. • false: Do not detect the image direction. Image rotation of 0, 90, 180, or 270 degrees counterclockwise can be recognized. The default value is false. That is, only images without rotation can be identified.

Response

Table 4-4 describes the response parameters.

Table 4-4 Response parameter description

Parameter	Type	Description
result	JSON	Invoking result when the invoking succeeds. The parameter is not included when the API invoking fails.
direction	Integer	Image direction. This parameter is valid only when detect_direction is set to true. The options are as follows: • 0: The image is not rotated. • 90: The image is rotated 90 degrees counterclockwise.

		• 180: The image is rotated 180 degrees counterclockwise. • 270: The image is rotated 270 degrees counterclockwise. When detect_direction is set to false, the value of this parameter is -1.
words_block_count	Integer	Number of text blocks to be recognized.
words_block_list	List	List of text blocks to be recognized. The output sequence is from left to right and from top to bottom.
words	String	Recognition result of a text block.
location	List	List of location information about a text block, including the 2D coordinates (x, y) of four vertexes in the text area, where the coordinate origin is the upper-left corner of the image, the x-axis is horizontal, and the y-axis is vertical.
error_code	String	Error code returned when the invoking fails. For details, see Error Codes. The parameter is not included when the API invoking succeeds.
error_msg	String	Error message returned when the API invoking fails. The parameter is not included when the API invoking succeeds.

Returned values

- Normal
- 200
- Failed

Table 4-5 Returned Value parameter description

Returned Value	Description
400	The request cannot be understood by the server due to malformed syntax. A client shall not submit the request again unless the request is modified. The request parameters are incorrect.
401	The request requires user authentication.
403	No permission to perform this operation.
404	The request failed because the requested resource could not be found on the server.
500	The server encountered an unexpected fault which prevented it from processing the request.

4.3.4 ID Card OCR API

Function overview

This API recognizes the text on an ID card image that has been uploaded (or the URL redirecting to the image file stored on OBS), and returns the result to the user.

Only the ID cards of the Chinese mainland can be recognized.

URI

URI format

POST /v1.0/ocr/id-card

Request

Table 4-6 describes the request parameters.

Table 4-6 Request parameter description

Parameter	Mandatory or Optional	Type	Description
image	Set either this parameter or url .	String	Image data, which is encoded based on Base64. The size of data encoded based on Base64 cannot exceed 10 MB. The pixels of each side of an image range from 15 to 8000. The supported image formats include JPG, PNG, BMP, and TIFF.
url	Set either this parameter or image .	String	URL of the image file. Currently, this URL can be accessed by temporarily authorization on HUAWEI CLOUD OBS or anonymous and public authorization.
side	Optional	String	• front: front of an ID card. • back: back of an ID card. If this parameter is left blank or not included, the system automatically recognizes whether the image is the front or back of an ID card. To improve accuracy, you are advised to set this parameter.

Response

Table 4-7 describes the response parameters.

Table 4-7 Response parameter description

Parameter	Type	Description
result	JSON	Invoking result when the invoking succeeds. The parameter is not included when the API invoking fails.
name	String	Name.

sex	String	Gender.
birth	String	Birthday.
ethnicity	String	Ethnicity.
address	String	Address.
number	String	ID card number.
issue	String	Authority that issues the ID card.
valid_from	String	Date from which an ID card takes effect.
valid_to	String	Date after which an ID card expires.
error_code	String	Error code returned when the invoking fails. For details, see Error Codes . The parameter is not included when the API invoking succeeds.
error_msg	String	Error message returned when the API invoking fails. The parameter is not included when the API invoking succeeds.

Returned values

- Normal
200
- Failed

Table 4-8 Returned Value parameter description

Returned Value	Description
400	The request cannot be understood by the server due to malformed syntax. A client shall not submit the request again unless the request is modified. The request parameters are incorrect.

401	The request requires user authentication.
403	No permission to perform this operation.
404	The request failed because the requested resource could not be found on the server.
500	The server encountered an unexpected fault which prevented it from processing the request.

4.3.5 VAT Invoice OCR API

Function overview

This API recognizes the text on a VAT invoice image that has been uploaded (or the URL redirecting to the image file stored on OBS), and returns the result to the user.

Only the VAT invoices in the People's Republic of China can be recognized.

URI

URI format

POST /v1.0/ocr/vat-invoice

Request

Table 4-9 describes the request parameters.

Table 4-9 Request parameter description

Parameter	Mandatory or Optional	Type	Description
image	Set either this parameter or url .	String	Image data, which is encoded based on Base64. The size of data encoded based on Base64 cannot exceed 10 MB. The image must be greater than or equal to 1000 x 500 pixels. The supported image formats include JPG, PNG, BMP, and TIFF.
url	Set either this parameter or	String	URL of the image file. Currently, this URL can be accessed by temporarily

	image.		authorization on HUAWEI CLOUD OBS or anonymous and public authorization.
--	---------------	--	--

Response

Table 4-10 describes the response parameters.

Table 4-10 Response parameter description

Parameter	Type	Description
result	JSON	Invoking result when the invoking succeeds. The parameter is not included when the API invoking fails.
type	String	VAT invoice type. The options are as follows: • special: special VAT invoice. • normal: normal VAT invoice. • electronic: electronic VAT invoice.
code	String	Invoice code.
check_code	String	Invoice verification code. If the verification code is not included in specific VAT invoices, an empty string is returned.
number	String	Invoice number.
issue_date	String	Date when an invoice is issued.
encryption_block	String	Password encryption block.
buyer_name	String	Buyer's name.
buyer_id	String	Buyer's taxpayer identifier.
buyer_address	String	Buyer's address and phone number.
buyer_bank	String	Buyer's account opening bank and bank account.
seller_name	String	Seller's name.

seller_id	String	Seller's taxpayer identifier.
seller_address	String	Seller's address and phone number.
seller_bank	String	Seller's account opening bank and bank account.
subtotal_amount	String	Amount in total.
subtotal_tax	String	Total tax amount.
total	String	Total price including tax.
item_list	List	List of goods or dutiable services.
name	String	Name of a commodity, dutiable service, or service.
specification	String	Specifications.
unit	String	Unit.
quantity	String	Quantity.
unit_price	String	Unit price.
amount	String	Amount of money.
tax_rate	String	Tax rate.
tax	String	Amount of tax.
error_code	String	Error code when the API invoking fails. The parameter is not included when the API invoking succeeds.
error_msg	String	Error message returned when the API invoking fails. The parameter is not included when the API invoking succeeds.

Returned values

- Normal

200

- Failed

Table 4-11 Returned Value parameter description

Returned Value	Description
400	The request cannot be understood by the server due to malformed syntax. A client shall not submit the request again unless the request is modified. The request parameters are incorrect.
401	The request requires user authentication.
403	No permission to perform this operation.
404	The request failed because the requested resource could not be found on the server.
500	The server encountered an unexpected fault which prevented it from processing the request.

4.4 Procedure

In this experiment, you need to download the SDK for OCR from the HUAWEI CLOUD service platform and use either of the following two methods to access the SDK. One method is to submit a RESTful service request by invoking the underlying APIs encapsulated by the SDK based on the AK/SK for identity authentication. The other method is to simulate the browser to submit a RESTful request by obtaining the user's token information. The procedure is as follows.

4.4.1 Applying for the Service

- Step 1 Log in to the HUAWEI CLOUD website and click Console in the upper right corner. The console page is displayed.**
- Step 2 Click Service List and choose EI Enterprise Intelligence > Optical Character Recognition. The OCR page is displayed.**

Step 3 On the Commercial Services tab page, click Apply for Service next to General Text OCR, ID Card OCR, and VAT Invoice OCR.

----End

4.4.2 Downloading the OCR SDK

Download the SDK software package and documents for OCR from the following website:

<https://developer.huaweicloud.com/en-us/sdk?OCR>

Step 1 Log in to the HUAWEI CLOUD website.

Open a browser, enter the following URL in the address box, and log in to the HUAWEI CLOUD website:

<https://www.huaweicloud.com/en-us/>

Step 2 Access the OCR service of HUAWEI CLOUD.

Choose Console > Service List > EI Enterprise Intelligence > OCR, as shown in the following figure.

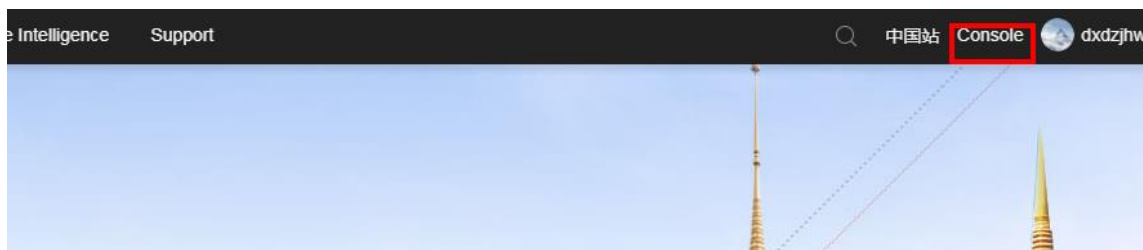


Figure 4-1 Console on the homepage

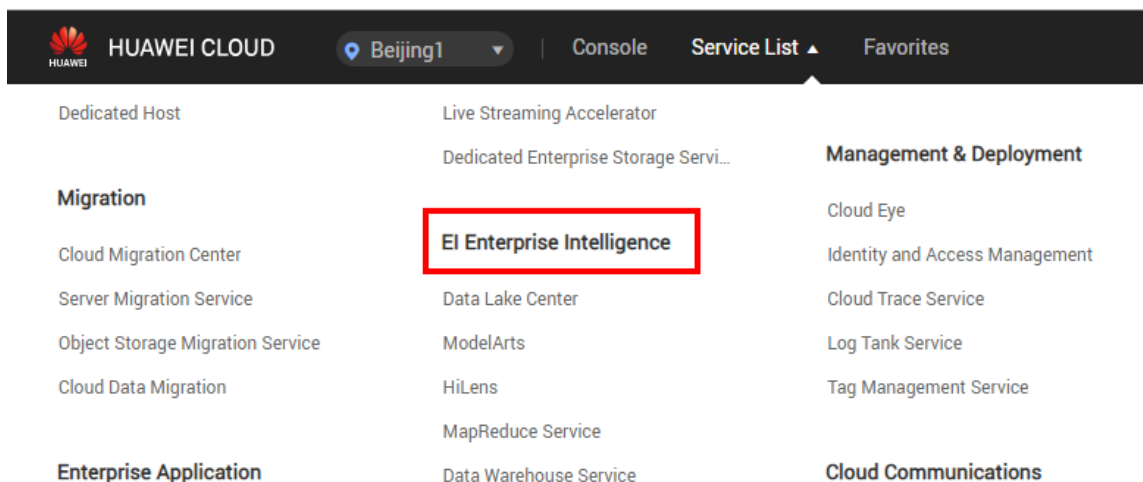


Figure 4-2 EI Enterprise Intelligence

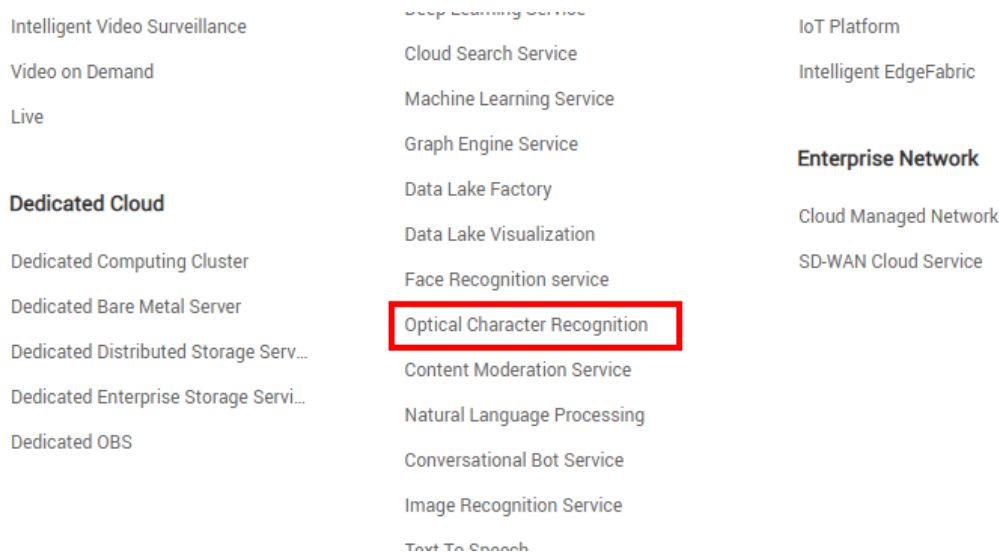


Figure 4-3 OCR

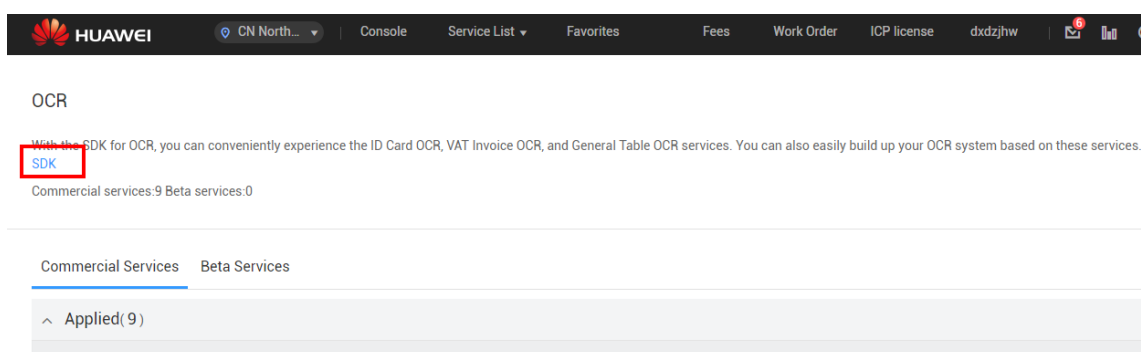


Figure 4-4 OCR

Step 3 Search for the OCR SDK.

Click the SDK link in Figure 4-4 to access the HUAWEI CLOUD SDK. The page shown in Figure 4-5 is displayed.

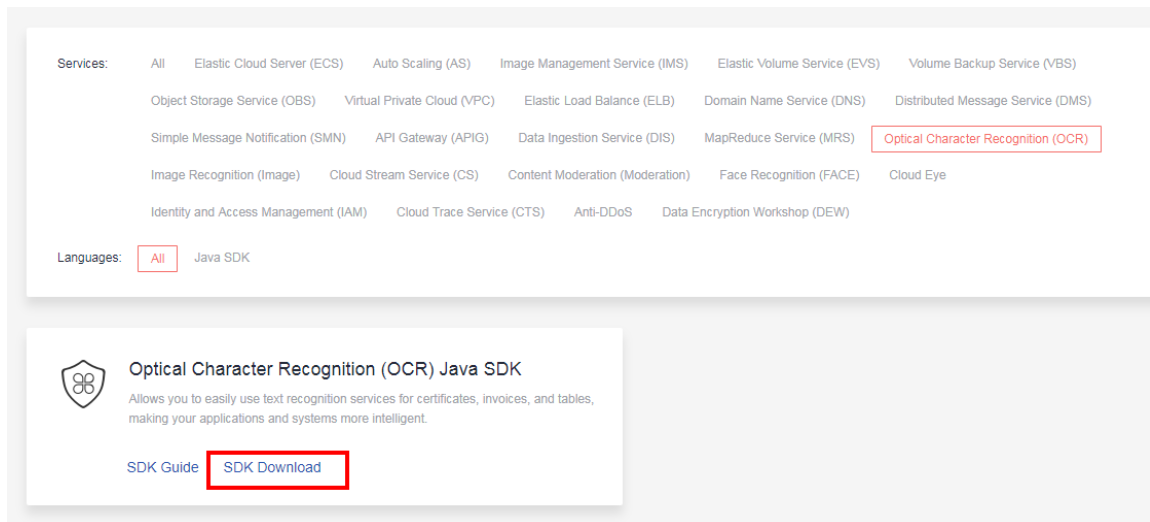


Figure 4-5 HUAWEI CLOUD SDK

Step 4 **Download the OCR SDK.**

Click **SDK Download** in the red box in the preceding figure to download the SDK file.

----End

4.4.3 Loading the SDK Project File

Step 1 **Decompress the SDK file and add it to the created workspace.**

Step 2 **Start the Eclipse.**

Step 3 **Ensure that the correct JRE path is configured under Windows > Preferences > Java > Installed JREs.**

Step 4 **Right-click Package Explorer on the left, choose Import from the shortcut menu, choose General > Existing Projects into Workspace, click Next, click Browse, and select the path where the OCR SDK is located.**

Step 5 **Click Finish to import the SDK, and open the project after the import succeeds. The project directory is shown in the following figure.**

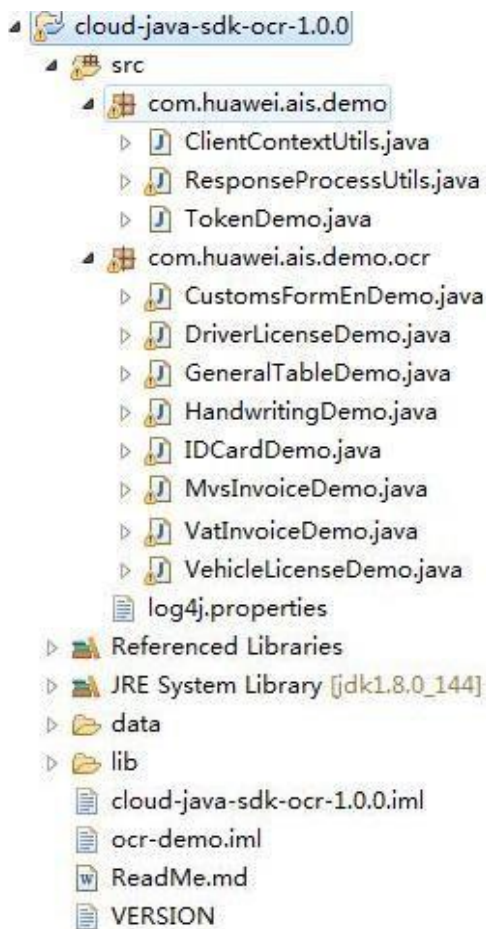


Figure 4-6 The Project Directory

----End

4.4.4 Using the General Text OCR Service Through the AK/SK

Step 1 Configure the access information for SDK user authentication.

1. Register an account and log in to the HUAWEI CLOUD console.
2. Move the cursor to the user name and select **My Credential** from the drop-down list box.
3. Click the **Access Keys** tab and click **Add Access Key**.
4. Enter the login password of the current user, use the email or SMS verification mode, and enter the verification code.

5. Click **OK** to download the AK/SK of the authentication account. Keep the AK/SK information secure. You can use the AK/SK information directly for subsequent experiments.
6. Enter the AK/SK of the specified user in **ClientContextUtils.java**, as shown in Figure 4-7.

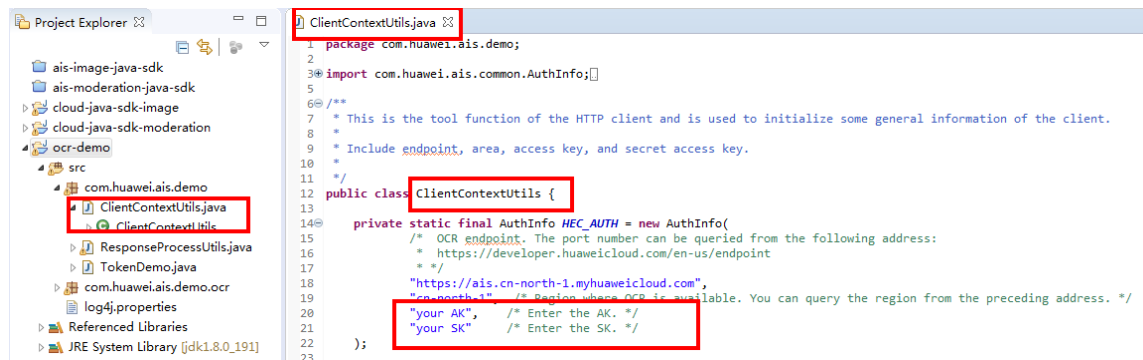


Figure 4-7 AK/SK configuration

Step 2 Select a local image or use the default image of the demo. Change the image path in the GeneralTextDemo.java file by referring to the following figure.

```
private static void generalTextDemo() throws IOException {
    //
    // 1. In the ClientContextUtils class, configure basic information for accessing AIS.
    // Generate a client connection object.
    //
    AisAccess service = new AisAccess(ClientContextUtils.getAuthInfo());

    //
    // 1.a AIS can be accessed in proxy mode. It applies to the scenario where a user cannot directly access HUAWEI CLOUD services,
    // If a proxy is used, you need to configure the parameter class (ProxyHostInfo) of the proxy server in ClientContextUtils.
    //
    //AisAccess service = new AisAccessWithProxy(ClientContextUtils.getAuthInfo(), ClientContextUtils.getProxyHost());

    try {
        //
        // 2. Construct the parameters required for accessing General Text OCR.
        //
        String uri = "/v1.0/ocr/general-text";
        byte[] fileData = FileUtils.readFileToByteArray(new File("data/general-text-demo.jpg"));
        String fileBase64Str = Base64.encodeBase64String(fileData);
    }
}
```

Figure 4-8 Image Path Modification Example

Step 3 Analyze the GeneralTextDemo.java class code.

```
package com.huawei.ais.demo.ocr;

import com.alibaba.fastjson.JSONObject;
import com.huawei.ais.demo.ClientContextUtils;
import com.huawei.ais.demo.ResponseProcessUtils;
import com.huawei.ais.sdk.AisAccess;
import com.huawei.ais.sdk.AisAccessWithProxy;
```

```
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.FileUtils;
import org.apache.http.HttpResponse;
import org.apache.http.entity.StringEntity;

import java.io.File;
import java.io.IOException;
import java.util.List;
import java.util.Map;

/**
 * Example class of using the general text OCR service.
 */
public class GeneralTextDemo {

    static {

    }

    //
    // Example function of using the general text OCR service.
    //
    /**
     *
     * @param uri
     * @param infos
     *     String : type,info
     *     type : image,url
     *     info : localPath,url
     * @throws IOException
     */
    private static void generalTextDemo() throws IOException {
        //
        // 1. In the ClientContextUtils class, configure basic information for
        // accessing the OCR service.
        // Generate a client connection object.
        //
        AisAccess service = new AisAccess(ClientContextUtils.getAuthInfo());

        //
        // 1.a The OCR service can be accessed in proxy mode. It applies to the
        // scenario where a user cannot directly access HUAWEI CLOUD services, for
        // example, when the intranet is used.
        // If a proxy is used, you need to configure the parameter class
        // (ProxyHostInfo) of the proxy server in ClientContextUtils.
        //
        //AisAccess service = new
        AisAccessWithProxy(ClientContextUtils.getAuthInfo(),
            ClientContextUtils.getProxyHost());

        try {
            //
            // 2. Initialize the parameters required for accessing the general
            // text OCR service.
            //
            String uri = "/v1.0/ocr/general-text";
```

```
//byte[] fileData = FileUtils.readFileToByteArray(new
File("data/general-text-demo.jpg"));

byte[] fileData = FileUtils.readFileToByteArray(new
File("data/1356924004_7530.jpg"));

String fileBase64Str = Base64.encodeBase64String(fileData);

JSONObject json = new JSONObject();
json.put("image", fileBase64Str);

//
// 2.a (Optional) Specify whether to check the text direction. If
this parameter is not specified, the text direction will not be detected by
default.
// The options of detect_direction are true and false.
//
json.put("detect_direction", false);

StringEntity stringEntity = new StringEntity(json.toJSONString(),
"utf-8");

// 3. Pass the URI parameter and required parameters of the OCR
service.
// Pass the parameters in JSON objects and invoke the service using
POST.
HttpResponse response = service.post(uri, stringEntity);

// 4. Check whether the service is invoked successfully. If 200 is
returned, the invoking succeeds. Otherwise, it fails.
ResponseProcessUtils.processResponseStatus(response);

// 5. Process the character stream returned by the service.
ResponseProcessUtils.processResponse(response);
} catch (Exception e) {
    e.printStackTrace();
} finally {

    // 6. Disconnect the client.
    service.close();
}
}

//
// Main entry point function.
//
public static void main(String[] args) throws IOException {
    // Test entry point function.
    generalTextDemo();
}
}
```

Step 4 Run the example program.

Right-click **GeneralTextDemo.java** in **Project Explorer**, choose **Run** from the menu bar, and click **Run As** to run the program. If **200** is displayed on the console, the program is successfully executed. The result of the general text OCR service can be displayed in the JSON editor (online JSON editor: <https://jsoneditoronline.org/>).

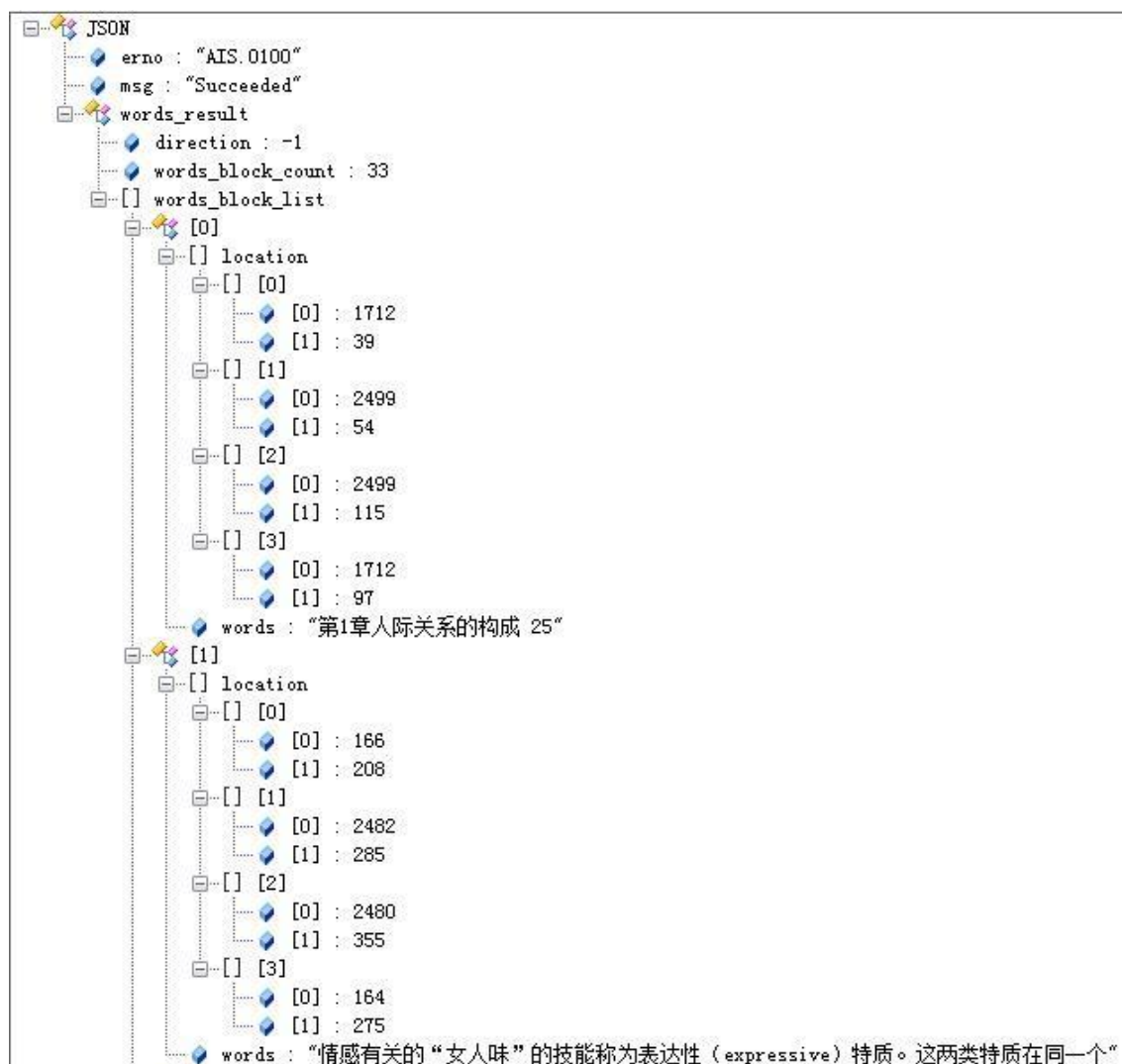


Figure 4-9 Program running result

NOTE: the API can be only used to detect Chinese characters, so the results contains Chinese characters.

Step 5 View the running result.

1. Response success example

```
{
  "result": {
    "direction": -1,
```

```
"words_block_count": 1,
"words_block_list": [
  {
    "words": "文字区域识别字母数字块 1",
    "location": [
      [15,15],
      [30,15],
      [30,30],
      [15,30]
    ]
  }
]
```

2. Response failure example

```
{
  "error_code": "AIS.0005",
  "error_msg": "The service does not exist."
}
```

----End

4.4.5 Using the ID Card OCR Service Through the AK/SK

- Step 1 **Configure the access information for SDK user authentication. The operations are the same as Step 1 for the general text OCR service.**
- Step 2 **Select a local image or use the default image of the demo. Change the image path in the IDCardDemo.java file by referring to the following figure.**

```
private static void idCardDemo() throws IOException {
    //
    // 1. In the ClientContextUtils class, configure basic information for accessing AIS.
    // Generate a client connection object.
    //
    AisAccess service = new AisAccess(ClientContextUtils.getAuthInfo());

    //
    // 1.a AIS can be accessed in proxy mode. It applies to the scenario where a user cannot directly access HUAWEI CLOUD services,
    // If a proxy is used, you need to configure the parameter class (ProxyHostInfo) of the proxy server in ClientContextUtils.
    //
    //AisAccess service = new AisAccessWithProxy(ClientContextUtils.getAuthInfo(), ClientContextUtils.getProxyHost());

    try {
        //
        // 2. Construct parameters required for accessing ID Card OCR.
        //
        String uri = "/v1.0/ocr/id-card";
        byte[] fileData = FileUtils.readFileToByteArray(new File("data/id-card-demo.png"));
        String fileBase64Str = Base64.encodeBase64String(fileData);

        JSONObject json = new JSONObject();
        json.put("image", fileBase64Str);

        //
        // 2.a (Optional) Specify whether the information is obtained from the front or back side of the ID card.
        // side Possible value: front, back, or unspecified. If the parameter is left unspecified or the parameter is not included,
        //
        //json.put("side", "front");
    }
}
```

Figure 4-10 Image Path Modification Example

Step 3 Analyze the IDCardDemo.java class code.

```
package com.huawei.ais.demo.ocr;

import com.alibaba.fastjson.JSONObject;
import com.huawei.ais.demo.ClientContextUtils;
import com.huawei.ais.demo.ResponseProcessUtils;
import com.huawei.ais.sdk.AisAccess;
import com.huawei.ais.sdk.AisAccessWithProxy;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.FileUtils;
import org.apache.http.HttpResponse;
import org.apache.http.entity.StringEntity;

import java.io.File;
import java.io.IOException;

/**
 * Example class of using the ID card OCR service.
 */
public class IDCardDemo {
    //
    // Example function of using the ID card OCR service.
    //
    private static void idCardDemo() throws IOException {
        //
        // 1. In the ClientContextUtils class, configure basic information for
        // accessing the OCR service.
        // Generate a client connection object.
        //
        AisAccess service = new AisAccess(ClientContextUtils.getAuthInfo());

        //
    }
}
```

```
// 1.a The OCR service can be accessed in proxy mode. It applies to the
scenario where a user cannot directly access HUAWEI CLOUD services, for
example, when the intranet is used.
// If a proxy is used, you need to configure the parameter class
(ProxyHostInfo) of the proxy server in ClientContextUtils.
//
//AisAccess service = new
AisAccessWithProxy(ClientContextUtils.getAuthInfo(),
ClientContextUtils.getProxyHost());

try {
    //
    // 2. Initialize the parameters required for accessing the ID card
OCR service.
    //
    String uri = "/v1.0/ocr/id-card";
    byte[] fileData = FileUtils.readFileToByteArray(new File("data/id-
card-demo.png"));
    String fileBase64Str = Base64.encodeBase64String(fileData);

    JSONObject json = new JSONObject();
    json.put("image", fileBase64Str);

    //
    // 2.a (Optional) Specify whether the information is obtained from
the front or back side of the ID card.
    // The options of side are front or back, and left blank. If the
parameter is left blank or not included, the algorithm automatically
determines the side from which the information is obtained. You are advised
to specify this parameter to improve the accuracy.
    //
    //json.put("side", "front");

    StringEntity stringEntity = new StringEntity(json.toJSONString(),
"utf-8");

    // 3. Pass the URI parameter and required parameters of the ID card
OCR service.
    // Pass the parameters in JSON objects and invoke the service using
POST.
    HttpResponse response = service.post(uri, stringEntity);

    // 4. Check whether the service is invoked successfully. If 200 is
returned, the invoking succeeds. Otherwise, it fails.
    ResponseProcessUtils.processResponseStatus(response);

    // 5. Process the character stream returned by the service.
    ResponseProcessUtils.processResponse(response);
} catch (Exception e) {
    e.printStackTrace();
} finally {

    // 6. Disconnect the client.
    service.close();
}
```

```

    }

    //
    // Main entry point function.
    //
    public static void main(String[] args) throws IOException {
        // Test entry point function.
        idCardDemo();
    }
}

```

Step 4 Run the example program.

Right-click **IDCardDemo.java** in **Project Explorer**, choose **Run** from the menu bar, and click **Run As** to run the program. If **200** is displayed on the console, the program is successfully executed. The result of the ID card OCR service can be displayed in the JSON editor.



Figure 4-11 Program running result

NOTE: the API can be only used to detect Chinese characters, so the results contains Chinese characters.

Step 5 View the running result.

1. Response success example (front)

```

{
    "result":{
        "name":"张三",
        "sex":"男",
        "ethnicity":"汉",
        "birth":"2000-03-06",
        "address":"XX 省 XX 市 XX 区 XX 街道 XX 号",
        "number":"610XXXXXXXXXXXXXXX"
    }
}

```

2. Response success example (back)

```
{
  "result":{
    "issue": "XX省XX市XX区XX派出所",
    "valid_from": "2004-08-26",
    "valid_to": "2034-08-26"
  }
}
```

3. Response failure example

```
{
  "error_code": "AIS.0005",
  "error_msg": "The service does not exist."
}
```

----End

4.4.6 Using the VAT Invoice OCR Service Through the AK/SK

Step 1 Configure the access information for SDK user authentication. The operations are the same as Step 1 for the general text OCR service.

Step 2 Select a local image or use the default image of the demo. Change the image path in the VatInvoiceDemo.java file by referring to the following figure.

```
private static void vatInvoiceDemo() throws IOException {
    //
    // 1. In the ClientContextUtils class, configure basic information for accessing AIS.
    // Generate a client connection object.
    //
    AisAccess service = new AisAccess(ClientContextUtils.getAuthInfo());

    //
    // 1.a AIS can be accessed in proxy mode. It applies to the scenario where a user cannot directly access HUAWEI CLOUD services,
    // If a proxy is used, you need to configure the parameter class (ProxyHostInfo) of the proxy server in ClientContextUtils.
    //
    //AisAccess service = new AisAccessWithProxy(ClientContextUtils.getAuthInfo(), ClientContextUtils.getProxyHost());

    try {
        //
        // 2. Construct parameters required for accessing VAT Invoice OCR.
        //
        String uri = "/v1.0/ocr/vat-invoice";
        byte[] fileData = FileUtils.readFileToByteArray(new File("data/vat-invoice-demo.jpg"));
        String fileBase64Str = Base64.encodeBase64String(fileData);
    }
}
```

Figure 4-12 Image Path Modification Example

Step 3 Analyze the VatInvoiceDemo.java class code.

```
package com.huawei.ais.demo.ocr;

import com.alibaba.fastjson.JSONObject;
import com.huawei.ais.demo.ClientContextUtils;
import com.huawei.ais.demo.ResponseProcessUtils;
import com.huawei.ais.sdk.AisAccess;
import com.huawei.ais.sdk.AisAccessWithProxy;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.FileUtils;
import org.apache.http.HttpResponse;
import org.apache.http.entity.StringEntity;

import java.io.File;
import java.io.IOException;

/**
 * Example class of using the VAT invoice OCR service.
 */
public class VatInvoiceDemo {
    //
    // Example function of using the VAT invoice OCR service.
    //
    private static void vatInvoiceDemo() throws IOException {
        //
        // 1. In the ClientContextUtils class, configure basic information for
        // accessing the service.
        // Generate a client connection object.
        //
        AisAccess service = new AisAccess(ClientContextUtils.getAuthInfo());

        //
        // 1.a The service can be accessed in proxy mode. It applies to the
        // scenario where a user cannot directly access HUAWEI CLOUD services, for
        // example, when the intranet is used.
        // If a proxy is used, you need to configure the parameter class
        // (ProxyHostInfo) of the proxy server in ClientContextUtils.
        //
        //AisAccess service = new
        AisAccessWithProxy(ClientContextUtils.getAuthInfo(),
            ClientContextUtils.getProxyHost());

        try {
            //
            // 2. Initialize the parameters required for accessing the VAT
            // invoice OCR service.
            //
            String uri = "/v1.0/ocr/vat-invoice";
            byte[] fileData = FileUtils.readFileToByteArray(new File("data/vat-
            invoice-demo.jpg"));
            String fileBase64Str = Base64.encodeBase64String(fileData);

            JSONObject json = new JSONObject();
            json.put("image", fileBase64Str);
            StringEntity stringEntity = new StringEntity(json.toJSONString(),
                "utf-8");
```

```
// 3. Pass the URI parameter and required parameters of the VAT
invoice OCR service.
// Pass the parameters in JSON objects and invoke the service using
POST.
HttpResponse response = service.post(uri, stringEntity);

// 4. Check whether the service is invoked successfully. If 200 is
returned, the invoking succeeds. Otherwise, it fails.
ResponseProcessUtils.processResponseStatus(response);

// 5. Process the character stream returned by the service.
ResponseProcessUtils.processResponse(response);
} catch (Exception e) {
    e.printStackTrace();
} finally {

    // 6. Disconnect the client.
    service.close();
}
}

//
// Main entry point function.
//
public static void main(String[] args) throws IOException {
    // Test entry point function.
    vatInvoiceDemo();
}
}
```

Step 4 Run the example program.

Right-click **VatInvoiceDemo.java** in **Project Explorer**, choose **Run** from the menu bar, and click **Run As** to run the program. If **200** is displayed on the console, the program is successfully executed. The result of the VAT invoice OCR service can be displayed in the JSON editor.



Figure 4-13 Program running result

NOTE: the API can be only used to detect Chinese characters, so the results contains Chinese characters.

Step 5 View the running result.

1. Response success example

```

{
  "result": {
    "type": "special",
    "code": "3103162166",
    "check_code": "",
    "number": "60543644",
    "issue_date": "2017 年 08 月 30 日",
    "encryption_block": "6/+1+733<672085+063>82>30<1872/1<>*312671<9<1-11208-746599*6/> +7>2163+141-8737*4932+7970*11892126>0*-+7+78>1",
    "buyer_name": "华为技术有限公司",
    "buyer_id": "917107277650880665",
    "buyer_address": "深圳市龙岗区坂田街道华为基地 400-830-8300",
    "buyer_bank": "CDE 银行 203845432526710",
    "seller_name": "餐饮有限公司",
  }
}

```

```
"seller_id": "9351099411892126",
"seller_address": "深圳龙岗区坂田街道 0755-82082000",
"seller_bank": "ABC 银行 9911015260873749",
"subtotal_amount": "¥8850.00",
"subtotal_tax": "¥1504.50",
"total": "¥10354.50",
"item_list": [
  {
    "name": "食品 A",
    "specification": "200g/盒",
    "unit": "盒",
    "quantity": "300",
    "unit_price": "28.00",
    "amount": "8400.00",
    "tax_rate": "17%",
    "tax": "1428.00"
  },
  {
    "name": "食品 B",
    "specification": "100g/袋",
    "unit": "袋",
    "quantity": "50",
    "unit_price": "9.00",
    "amount": "450.00",
    "tax_rate": "17%",
    "tax": "76.50"
  }
]
}
}
```

2. Response failure example

```
{
  "error_code": "AIS.0005",
  "error_msg": "The service does not exist."
}
```

----End

4.5 Summary

This section describes how to use the HUAWEI CLOUD OCR service for experiment operations. The SDK is used to send RESTful requests to implement related functions. When using the SDK to initiate a RESTful request, configure the necessary user authentication information. Two authentication modes are available: AK/SK and token. This section describes the AK/SK systematically, providing an operation guide for using the general text OCR, ID card OCR, and VAR invoice OCR services.

5 Face Recognition

5.1 Experiment Introduction

Face recognition is an intelligent service that uses a computer to process, analyze, and understand face images based on human facial features. Face recognition is available through open APIs. You can obtain the facial processing results by accessing and invoking the APIs in real time. The APIs can automatically recognize and compare faces, query the similarity, and build an intelligent service system, thereby improving service efficiency.

Face detection accurately locates faces in images and identifies the size of each face. It can also detect front faces and side faces with different tilts in images. This sub-service is a basic service in the face recognition field and is applicable to various application scenarios such as security protection, electronic identity, and criminal investigation of public security.

Face verification returns the similarity degree between two faces in two images by comparing the facial features. If the two images contain multiple faces, the system compares the similarity between the largest faces in the two images.

This service can release the POST RESTful HTTP request service in two modes. One is to invoke the underlying APIs encapsulated by the SDK, and the other is to simulate the access to the server from the front-end browser. The first mode needs to use the user's AK/SK for face recognition, and the second mode needs to obtain the user's token for face recognition. In this experiment, the AK/SK authentication mode is used to request a service.

5.2 Experiment Objectives

This experiment describes how to use the face recognition function provided by HUAWEI CLOUD services and guides trainees to detect and compare faces. Currently, HUAWEI CLOUD provides the RESTful API for face recognition. Through this experiment,

trainees can understand and master how to use Java to develop the face recognition service.

5.3 API Introduction

5.3.1 RESTful API Introduction

HUAWEI CLOUD APIs comply with RESTful API design specifications. Representational State Transfer (REST) allocates Uniform Resource Identifiers (URIs) to dispersed resources so that the resources can be located. Applications on clients use Uniform Resource Locators (URLs) to obtain the resources.

5.3.2 RESTful API Request/Response Composition

A RESTful API request/response consists of the following five parts:

- Request URL

The URL format is as follows: `https://Endpoint/uri`. The parameters in the URL are described in Table 5-1.

Table 5-1 URL parameter description

Parameter	Description
Endpoint	Web service entrance URL. Obtain this value from Regions and Endpoints . Endpoint face.cn-north-1.myhuaweicloud.com corresponding to face recognition is used by all service APIs.
uri	Resource path, that is, API access path. Obtain the value from the URI of a specified API, for example, <code>/v1.0/ais/subscribe</code> .

- Request header

The request header consists of two parts: HTTP method and optional additional request header field (such as the field required by a specified URI and HTTP method).

Table 5-2 describes the request methods supported by RESTful APIs.

Table 5-2 Request method description

Method	Description
GET	Requests the server to return specified resources.
PUT	Requests the server to update specified resources.
POST	Requests the server to add resources or perform a special operation.
DELETE	Requests the server to delete specified resources, for example, objects.
PATCH	Requests the server to update partial content of a specified resource. If a target resource does not exist, PATCH may create a resource.

For details about common request headers, see https://support.huaweicloud.com/en-us/api-face/face_02_0027.html.

- Request body

A request body is generally sent in a structured format (for example, JSON or XML), corresponding to Content-type in the request header, and is used to transfer content except the request header. If a request body contains a parameter in Chinese, the parameter must be coded in UTF-8 mode.

- Response header

A response header contains two parts: status code and additional response header field.

Status code, including success codes 2xx and error codes 4xx or 5xx. For details about the status code, see https://support.huaweicloud.com/en-us/api-face/face_02_0027.html. Additional response header field, such as the field required by the response supporting a request (the field in the Content-type response header). For details, see https://support.huaweicloud.com/en-us/api-asr/asr_03_0013.html.

- Response body

A response body is generally returned in a structured format (for example, JSON or XML), and is used to transfer content except the response header. When a service request is successfully sent, the service request result is returned. When a service

request fails to be sent, an error code is returned. For details, see https://support.huaweicloud.com/en-us/api-asr/asr_03_0014.html.

5.3.3 Request Initiation Methods

There are three methods to initiate constructed requests, including:

- cURL

cURL is a command line tool, which can be used to perform URL operations and transfer information. cURL functions as an HTTP client to send HTTP requests to the server and receive responses. cURL is applicable to API debugging. For details about cURL, see <https://curl.haxx.se/>.

- Code

You can invoke APIs by coding to assemble, send, and process requests.

- Mozilla and Google provide graphical browser plug-ins for REST clients to send and process requests.

For Firefox, see [Firefox RESTClient](#).

For Chrome, see [Postman Interceptor](#).

5.3.4 Obtaining the Project ID

The **project_id** parameter is a mandatory parameter for invoking the HUAWEI CLOUD face recognition service.

Step 1 Log in to the HUAWEI CLOUD website and click Console.

Step 2 Move the cursor to the user name in the upper right corner and select My Credential from the drop-down list box.

Step 3 On the My Credential page, view the project ID in the project list.

Projects		Access Keys
Enter a project name.		
Region	Project Name	Project ID
AP-Hong Kong	ap-southeast-1	05ba1ebadfd344e1b1f0a...
AP-Bangkok	ap-southeast-2	10561bb3b3fa48f0a...
CN East-Shanghai2	cn-east-2	5898b13f9ef84ab49b4f0a...
CN North-Beijing1	cn-north-1	16ae80a6a67f0a...
CN North-Beijing4	cn-north-4	93d81fb01bf74b34f0a...
CN South-Guangzhou	cn-south-1	059a5913239441e7f0a...

Figure 5-1 Project ID

Step 4 **Copy the project ID, which is the value of project_id of the area to which the face recognition service belongs.**

----End

5.3.5 Face Detection API

Function overview

This API detects and analyzes faces in images.

URI

URI format

POST /v1/{project_id}/face-detect

Request

Table 5-3 describes the request parameters.

Table 5-3 Request parameter description

Parameter	Mandatory or Optional	Data Type	Description
image_url	Select one from this parameter, image_file , and image_base64 .	String	Image URL. Currently, only temporarily authorized URLs redirecting to image files stored on HUAWEI CLOUD OBS are supported.
image_file	Select one from this parameter, image_url , and image_base64 .	File	Local image file. The request format is multipart when a file is being uploaded. The image size cannot exceed 12 MB.

image_base64	Select one from this parameter, image_file , and image_url .	String	Image data, which is encoded based on Base64. The following requirements must be met: • The size of data encoded based on Base64 cannot exceed 12 MB. • The supported image formats include JPG, JPEG, BMP, and PNG.
attributes	Optional	String	Indicates whether to return the facial attribute list. Multiple attributes are separated by commas (,). Currently, the following attributes are supported: • 0: face posture • 1: gender • 2: age • 3: key facial characteristic • 4: dress (hat and glasses) • 5: smiling face

Response

Table 5-4 describes the response parameters.

Table 5-4 Response parameter description

Parameter	Data Type	Description
faces	List [DetectFace]	Detected face. For details about the DetectFace structure, see DetectFace . The parameter is not included when the API invoking fails.
error_code	String	Error code returned when the invoking

		fails. For details, see Error Codes . The parameter is not included when the API invoking succeeds.
error_msg	String	Error message returned when the API invoking fails. The parameter is not included when the API invoking succeeds.

Returned Value

- Normal

200

- Failed

Table 5-5 Returned Value parameter description

Returned Value	Description
400	The request cannot be understood by the server due to malformed syntax. A client shall not submit the request again unless the request is modified. The request parameters are incorrect.
401	The request requires user authentication.
403	No permission to perform this operation.
404	The request failed because the requested resource could not be found on the server.
500	The server encountered an unexpected fault which prevented it from processing the request.

5.3.6 Face Verification API

Function overview

This API compares two faces to check whether they are the same person and returns the confidence score.

If each input image contains multiple faces, the API selects the largest face for comparison.

URI

URI format

POST /v1/{project_id}/face-compare

Request

Table 5-6 describes the request parameters.

Table 5-6 Request parameter description

Parameter	Mandatory or Optional	Data Type	Description
image1_url	Select one from this parameter, image1_file , and image1_base64 .	String	Image URL. Currently, only temporarily authorized URLs redirecting to image files stored on HUAWEI CLOUD OBS are supported.
image1_file	Select one from this parameter, image1_url , and image1_base64 .	File	Local image file. The request format is multipart when a file is being uploaded. The image size cannot exceed 12 MB.
image1_base64	Select one from this parameter, image1_file , and image1_url .	String	Image data, which is encoded based on Base64. The following requirements must be met: - The size of data encoded based on Base64 cannot exceed 12 MB. - The supported image formats include JPG, JPEG, BMP, and PNG.
image2_url	Select one from this parameter, image2_file , and image2_base64 .	String	Image URL. Currently, only temporarily authorized URLs redirecting to image files stored on HUAWEI CLOUD OBS are

			supported.
image2_file	Select one from this parameter, image2_url , and image2_base64 .	File	Local image file. The request format is multipart when a file is being uploaded. The image size cannot exceed 12 MB.
image2_base64	Select one from this parameter, image2_file , and image2_url .	String	Image data, which is encoded based on Base64. The following requirements must be met: - The size of data encoded based on Base64 cannot exceed 12 MB. - The supported image formats include JPG, JPEG, BMP, and PNG.

Response

Table 5-7 describes the response parameters.

Table 5-7 Response parameter description

Parameter	Data Type	Description
image1_face	DetectFace	Face detected in the first image. For details about the DetectFace structure, see DetectFace . The parameter is not included when the API invoking fails.
image2_face	DetectFace	Face detected in the second image. The parameter is not included when the API invoking fails.
similarity	Double	Face similarity. The value ranges from 0 to 1. A larger value indicates a higher similarity degree. Generally, if the value is greater than 0.93 , it is regarded that the faces in two

		images belong to one person. The parameter is not included when the API invoking fails.
error_code	String	Error code returned when the invoking fails. For details, see Error Codes. The parameter is not included when the API invoking succeeds.
error_msg	String	Error message returned when the API invoking fails. The parameter is not included when the API invoking succeeds.

Returned values

- Normal

200

- Failed

Table 5-8 Returned Value parameter description

Returned Value	Description
400	The request cannot be understood by the server due to malformed syntax. A client shall not submit the request again unless the request is modified. The request parameters are incorrect.
401	The request requires user authentication.
403	No permission to perform this operation.
404	The request failed because the requested resource could not be found on the server.
500	The server encountered an unexpected fault which prevented it from processing the request.

5.4 Procedure

In this experiment, you need to download the SDK for face recognition from the HUAWEI CLOUD service platform and use either of the following two methods to access the SDK. One method is to submit a RESTful service request by invoking the underlying APIs encapsulated by the SDK based on the AK/SK for identity authentication. The other method is to simulate the browser to submit a RESTful request by obtaining the user's token information. The procedure is as follows.

5.4.1 Applying for the Service

- Step 1 **Log in to the HUAWEI CLOUD website and click Console in the upper right corner. The console page is displayed.**
- Step 2 **Click Service List and choose EI Enterprise Intelligence > Face Recognition service. The Face Recognition page is displayed.**
- Step 3 **On the Service tab page, apply for the Face Detection and Face Verification services.**

----End

5.4.2 Using the Face Detection Service Through the AK/SK

- Step 1 **Configure the access information for SDK user authentication.**
 - 1. Register an account and log in to the HUAWEI CLOUD console.
 - 2. Move the cursor to the user name and select **My Credential** from the drop-down list box.
 - 3. Click the **Access Keys** tab and click **Add Access Key**.
 - 4. Enter the login password of the current user, use the email or SMS verification mode, and enter the verification code.
 - 5. Click **OK** to download the AK/SK of the authentication account. Keep the AK/SK information secure. You can use the AK/SK information directly for subsequent experiments.
 - 6. Enter the AK/SK of the specified user in **ClientContextUtils.java**, as shown in Figure 5-2.

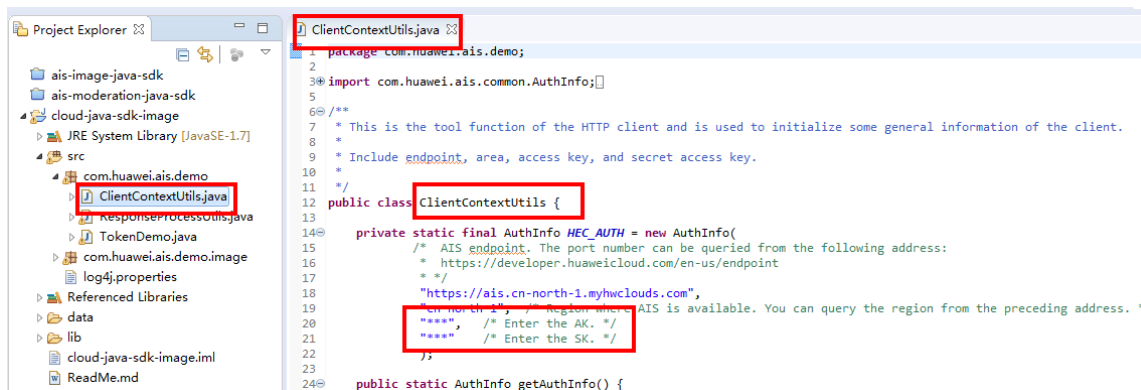


Figure 5-2 AK/SK configuration

Step 2 Add the FaceDetectDemo.java class to the com.huawei.ais.demo.image package in the image recognition project (ais-image-java-sdk) to invoke the face detection service.

Analysis of the FaceDetectDemo.java code:

```
package com.huawei.ais.demo.image;
import java.io.File;
import java.io.IOException;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.FileUtils;
import com.huawei.ais.service.AccessService;
import com.huawei.ais.service.AccessServiceImpl;

/**
 * Detect and analyze faces in images.
 * @author
 *
 */
public class FaceDetectDemo {

    private static void faceDetectDemo() {
        // 1. Configure basic information for accessing the face detection
        service.
        String serviceName = "serviceName";
        String region= "regionName";

        // TODO: Enter the user's AK/SK information.
        String ak = "HE4WYAKKJM59V*****";
        String sk = "pXmBN2paGSr8bgdFBcQMyjXATlb*****G4";
        AccessService service = new AccessServiceImpl(serviceName, region, ak,
        sk);
    }
}
```

```
try {
    // 2. Use the GET method to check whether the face detection service
    is enabled.
    String url = "https://face.cn-north-
1.myhuaweicloud.com/v1/7c3470205a694cc9994671ce02fca48a/subscribe";// Used to
query the service provisioning status.
    service.get(ak, sk, url);

    // 3. Create an API for face detection requests.
    String postUrl = "https://face.cn-north-
1.myhuaweicloud.com/v1/7c3470205a694cc999467****/face-detect";
    // 4. Enter the address of the image to be detected.
    byte[] fileData = FileUtils.readFileToByteArray(new
File("data/20181102-141433(eSpace).png"));
    String fileBase64Str = Base64.encodeBase64String(fileData);
    String postbody = "{\"image_base64\": \"" + fileBase64Str +
"\", \"attributes\": \"0,1,2,3,4,5\"}";

    // 5. Invoke the service by using the POST method to detect and
analyze faces in images.
    service.post(ak, sk, postUrl, postbody);

} catch (IOException e) {
    // 6. Disconnect the client.
    service.close();
}

// Main entry point function.
public static void main(String[] args) throws IOException {
    // Test entry point function.
    faceDetectDemo();
}
}
```

Step 3 **Add the AccessService and AccessServiceImpl classes to the project to implement API gateway communication.**

To view the source code, log in to the HUAWEI CLOUD website, and choose Support > Documentation > Enterprise Intelligence > Face Recognition > API Reference > Environment Preparation > Authenticating API Requests > AK/SK Authentication. (The link is https://support.huaweicloud.com/en-us/api-face/face_02_0005.html.)

In this experiment, the AccessService class and the AccessServiceImpl class are encapsulated into a jar file. Perform the following operations:

Copy the faceapi.jar file provided for this experiment to the lib folder of the project.

(Download link:

<https://data-certification.obs.cn-east-2.myhuaweicloud.com/ENG/HCIP-AI%20EI%20Developer/V1.0/faceapi.jar>)

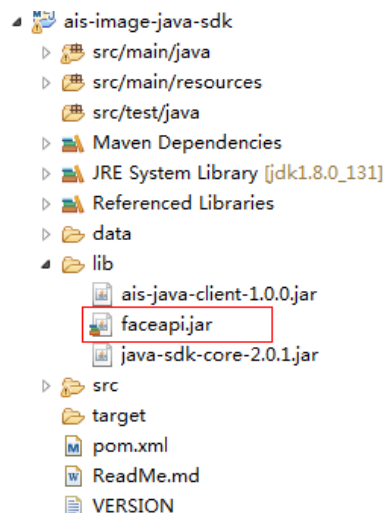


Figure 5-3 The Project Directory

Right-click the .jar file and choose **Build Path** from the shortcut menu. Then, click **Add to Build Path**.

Step 4 Run the example program.

Right-click **FaceDetectDemo.java** in **Project Explorer**, choose **Run** from the menu bar, and click **Run As** to run the program. If **200** is displayed on the console, the program is successfully executed.

Step 5 View the running result.

1. Response success example

```
{
  "faces": [
    {
      "bounding_box": {
        "width": 174,
        "top_left_y": 37,
        "top_left_x": 22,
        "height": 174
      },
    },
  ],
}
```

```
"attributes": {
  "age": 35,
  "smile": "yes",
  "gender": "male",
  "dress": {
    "glass": "none",
    "hat": "none"
  },
  "headpose": {
    "pitch_angle": -3.8639126,
    "roll_angle": -3.988193,
    "yaw_angle": -1.0292832
  }
}
}
```

2. Response failure example

```
{
  "error_code": "AIS.0005",
  "error_msg": "The service does not exist."
}
```

----End

5.4.3 Using the Face Verification Service Through the AK/SK

Step 1 **Configure the access information for SDK user authentication.**

1. Register an account and log in to the HUAWEI CLOUD console.
2. Move the cursor to the user name and select **My Credential** from the drop-down list box.
3. Click the **Access Keys** tab and click **Add Access Key**.
4. Enter the login password of the current user, use the email or SMS verification mode, and enter the verification code.

5. Click **OK** to download the AK/SK of the authentication account. Keep the AK/SK information secure. You can use the AK/SK information directly for subsequent experiments.
6. Enter the AK/SK of the specified user in **ClientContextUtils.java**, as shown in Figure 5-4.

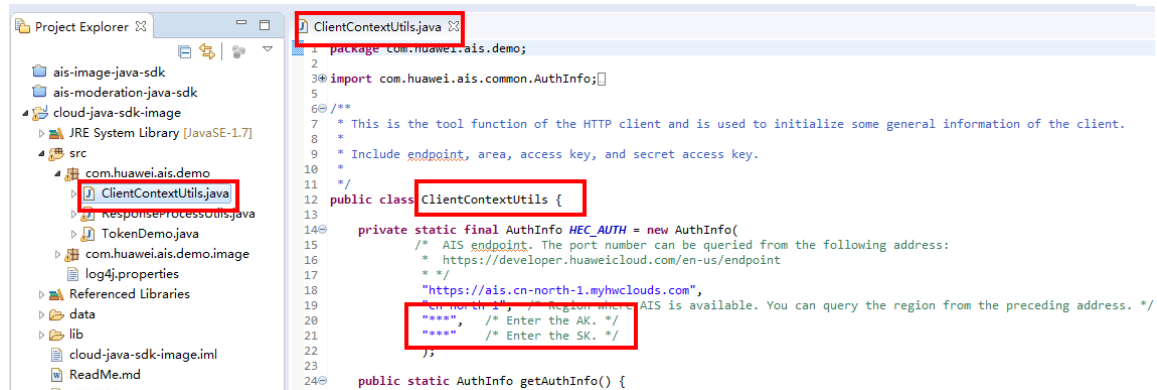


Figure 5-4 AK/SK configuration

Step 2 Add the FaceCompareDemo.java class to the project to invoke the face verification service.

Analysis of the **FaceCompareDemo.java** code:

```
package com.huawei.ais.demo.image;
import java.io.File;
import java.io.IOException;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.FileUtils;
import com.huawei.ais.service.AccessService;
import com.huawei.ais.service.AccessServiceImpl;

/**
 * Compare two faces to check whether they are the same person and return the
 * confidence score.
 * If each input image contains multiple faces, the API selects the largest
 * face for comparison.
 * @author
 *
 */
public class FaceCompareDemo {

    private static void faceCompareDemo() {

        // 1. Configure basic information for accessing the face verification
        service.
        String serviceName = "serviceName";
        String region= "regionName";
```

```
// TODO: Enter the user's AK/SK information.
String ak = "HE4WYAKKJM59N*****";
String sk = "pXmBN2paGSr8bgdFBcQMyjXATlXkb*****G4";
AccessService service = new AccessServiceImpl(serviceName, region, ak,
sk);

try {
    // 2. Use the GET method to check whether the face verification
service is enabled.
    String url = "https://face.cn-north-
1.myhuaweicloud.com/v1/7c3470205a694cc9994671ce02fca48a/subscribe";// Used to
query the service provisioning status.
    service.get(ak, sk, url);

    // 3. Create an API for face verification requests.
    String postUrl = "https://face.cn-north-
1.myhuaweicloud.com/v1/7c3470205a694cc999467****/face-compare";
    // 4. Enter the addresses of the images to be compared.
    byte[] fileData1 = FileUtils.readFileToByteArray(new
File("data/20181102-141241(eSpace).jpg"));
    byte[] fileData2 = FileUtils.readFileToByteArray(new
File("data/20181102-141328(eSpace).jpg"));
    String fileBase64Str1 = Base64.encodeBase64String(fileData1);
    String fileBase64Str2 = Base64.encodeBase64String(fileData2);
    String postbody = "{\"image1_base64\":\"" + fileBase64Str1 +
"\",\"image2_base64\":\"" + fileBase64Str2 + "\"}";

    // 5. Invoke the service by using the POST method to compare faces in
images.
    service.post(ak, sk, postUrl, postbody);

} catch (IOException e) {
    // 6. Disconnect the client.
    service.close();
}

}

// Main entry point function.
public static void main(String[] args) throws IOException {
    // Test entry point function.
    faceCompareDemo();
}

}
```

Step 3 Run the example program.

Right-click **FaceCompareDemo.java** in **Project Explorer**, choose **Run** from the menu bar, and click **Run As** to run the program. If **200** is displayed on the console, the program is successfully executed.

Step 4 View the running result.

1. Response success example

```
{
  "image1_face": {
    "bounding_box": {
      "width": 174,
      "top_left_y": 37,
      "top_left_x": 22,
      "height": 174
    }
  },
  "similarity": 0.4078676104545593,
  "image2_face": {
    "bounding_box": {
      "width": 118,
      "top_left_y": 28,
      "top_left_x": 94,
      "height": 118
    }
  }
}
```

2. Response failure example

```
{
  "error_code": "AIS.0005",
  "error_msg": "The service does not exist."
}
```

----End

5.5 Summary

This section describes how to use the HUAWEI CLOUD face recognition service for experiment operations. The SDK is used to send RESTful requests to implement related functions. When using the SDK to initiate a RESTful request, configure the necessary user authentication information. Two authentication modes are available: AK/SK and token. This section describes the AK/SK systematically, providing an operation guide for using the face recognition service for face detection and verification.



Huawei HCIP-AI Certification Training

HCIP- AI EI Developer

Speech Processing Experiment

Guide

ISSUE:1.0



HUAWEI TECHNOLOGIES CO., LTD.

Copyright © Huawei Technologies Co., Ltd. 2019. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <http://e.huawei.com>

Huawei Certificate System

Huawei Certification follows the "platform + ecosystem" development strategy, which is a new collaborative architecture of ICT infrastructure based on "Cloud-Pipe-Terminal". Huawei has set up a complete certification system consisting of three categories: ICT infrastructure certification, Platform and Service certification and ICT vertical certification, and grants Huawei certification the only all-range technical certification in the industry.

Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

The fourth revolution of science and technology, driven by ICT networks and AI, is leading us towards an intelligent world where all things can sense, all things are connected, and all things are intelligent. In October 2018, Huawei Rotating Chairman Eric Xu announced Huawei's AI strategy, as well as its full-stack, all-scenario AI portfolio. The company's full-stack, all-scenario AI portfolio is designed to provide powerful support for Huawei Cloud EI and HiAI. Backed by its AI portfolio, Huawei Cloud EI will be able to deliver a full-stack portfolio for enterprise and government customers, and HiAI will provide a full-stack portfolio for smart devices. HiAI services are deployed on Huawei Cloud EI. However, the gap of AI talent in this world is very large. To seize this historic opportunity and accelerate the construction of the AI talent ecosystem, we hereby release the HCIP-AI EI Developer V1.0 certification to develop talents who have acquired basic theoretical knowledge about image processing, speech processing, and natural language processing (NLP) and who are able to use Huawei enterprise AI solutions (such as Huawei cloud EI) and general open-source frameworks to conduct development and innovation.

HCIP-AI EI Developer V1.0 is a kind of engineer career certification for ICT practitioners, Huawei's users, Huawei engineers, Huawei partners' engineers, and college students. The content of HCIP-AI EI Developer V1.0 includes but is not limited to: deep feedforward networks, image processing theory and applications, speech processing theory and applications, NLP theory and applications, the image processing experiment, the speech processing experiment, and the NLP experiment.

About This Document

Overview

This document is an HCIP-AI certification training course. It is intended for trainees who are preparing for HCIP-AI tests or readers who want to know about AI basics. After understanding this document, you will be able to perform speech processing, for example, speech file pre-processing, speech input, text to speech (TTS), and automatic speech recognition (ASR), and carry out development. To implement the ASR and TTS operations, we use the tensorflow framework to construct the deep neural network, such as BiRNN and Seq2Seq model.

Description

This document contains three experiments and it involves speech file pre-processing, Tensorflow-based TTS and ASR. It aims to improve the practical development capability of AI speech processing.

- Experiment 1 helps understand Python-based speech file pre-processing.
- Experiment 2 helps understand Tensorflow-based TTS.
- Experiment 3 helps understand Tensorflow-based ASR.

Background Knowledge Required

This course is designed for development training for Huawei certification. To better understand content of this document, readers are required to meet the following basic conditions.

- Have basic Python and Java language programming capabilities.
- Have basic knowledge in data structure.
- Have basic knowledge in tensorflow.
- Have basic knowledge in deep neural network.

Experiment Environment Overview

- Windows (64-bit)
- Anaconda3 (64-bit) (Python 3.6.4 or later)
- Jupyter Notebook
- Link for downloading the experiment data set:
 - (1) Speech Pre-processing: <https://data-certification.obs.cn-east-2.myhuaweicloud.com/ENG/HCIP-AI%20EI%20Developer/V1.0/Speech%20Pre-processing.zip>
 - (2) TTS: <https://data-certification.obs.cn-east-2.myhuaweicloud.com/ENG/HCIP-AI%20EI%20Developer/V1.0/TTS.zip>
 - (3) ASR: <https://data-certification.obs.cn-east-2.myhuaweicloud.com/ENG/HCIP-AI%20EI%20Developer/V1.0/ASR.zip>

Contents

About This Document.....	2
Overview	2
Description	2
Background Knowledge Required	2
Experiment Environment Overview	3
1 Speech Pre-processing	5
1.1 Introduction to the Experiment	5
1.2 Purpose	5
1.3 Knowledge Required	6
1.4 Installing Related Modules.....	6
1.5 Procedure	7
1.6 Summary	15
2 TTS.....	16
2.1 Introduction to the Experiment.....	16
2.2 Purpose	16
2.3 Knowledge Required	17
2.4 Procedure	17
2.4.1 Build Basic Modules.....	17
2.4.2 Build The Whole Training Process	30
2.5 Summary	38
3 ASR	39
3.1 Introduction to the Experiment.....	39
3.2 Purpose	39
3.3 Knowledge Required	40
3.4 Procedure	40
3.4.1 Build Basic Modules.....	40
3.4.2 Build The Whole Training Process	47
3.5 Summary	52

1 Speech Pre-processing

1.1 Introduction to the Experiment

Speech is a non-stationary time-varying signal. It carries various information. Information included in speech needs to be extracted for speech processing, for example, speech encoding, TTS, speech recognition, and speech quality enhancement. Generally, speech data is processed to analyze speech signals and extract characteristic parameters for subsequent processing or to process speech signals. For example, background noise is suppressed in speech quality enhancement to obtain relatively "clean" speech. In TTS, splicing and smoothing need to be performed for speech segments to obtain synthetic speech with higher subjective speech quality. Applications in this aspect are also created on the basis of analysis and extraction of speech signal information. In a word, the purpose of speech signal analysis is to conveniently and effectively extract and express information carried in speech signals. Based on types of analyzed parameters, speech signal analysis can be divided into time-domain analysis and transform-domain (frequency domain and cepstral domain) analysis. The time-domain analysis method is the simplest and the most intuitive method. It directly analyzes time-domain waveforms of speech signals and extracts characteristic parameters, including short-time energy and average amplitude of speech, average short-time zero-crossing rate, short-time autocorrelation function, and short-time average amplitude difference function.

This experiment provides analysis based on speech data attributes of the short-sequence speech data set and related characteristic attributes to have a more in-depth and comprehensive understanding of speech data.

1.2 Purpose

Based on the Python language and switch segment sequence speech data set, this experiment intuitively checks speech data attributes by using a third-party framework to enable trainees to have a more in-depth understanding of speech data modes.

1.3 Knowledge Required

This experiment requires knowledge in two aspects:

- Syntactical basis of the Python language and hands-on operation capability
- Understanding of the related wave framework

1.4 Installing Related Modules

Install the Python module.

Click **Start** in the lower left corner of the Windows OS. A menu list is displayed.

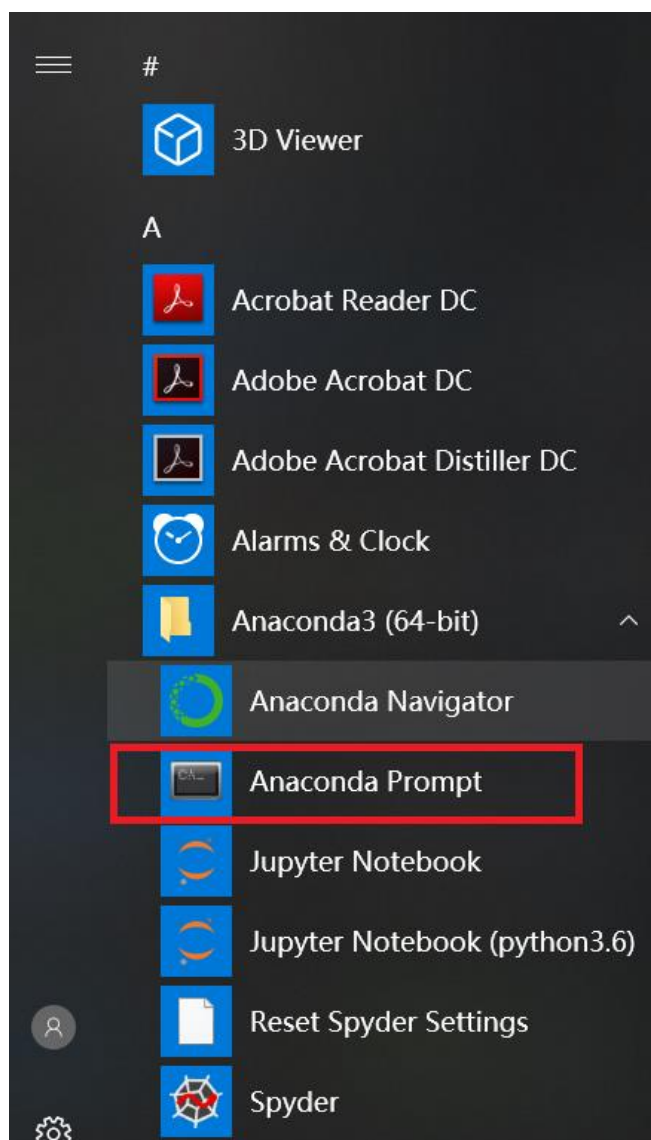


Figure 1-1 Anaconda Prompt

Click **Anaconda Prompt**. The Anaconda system is displayed.

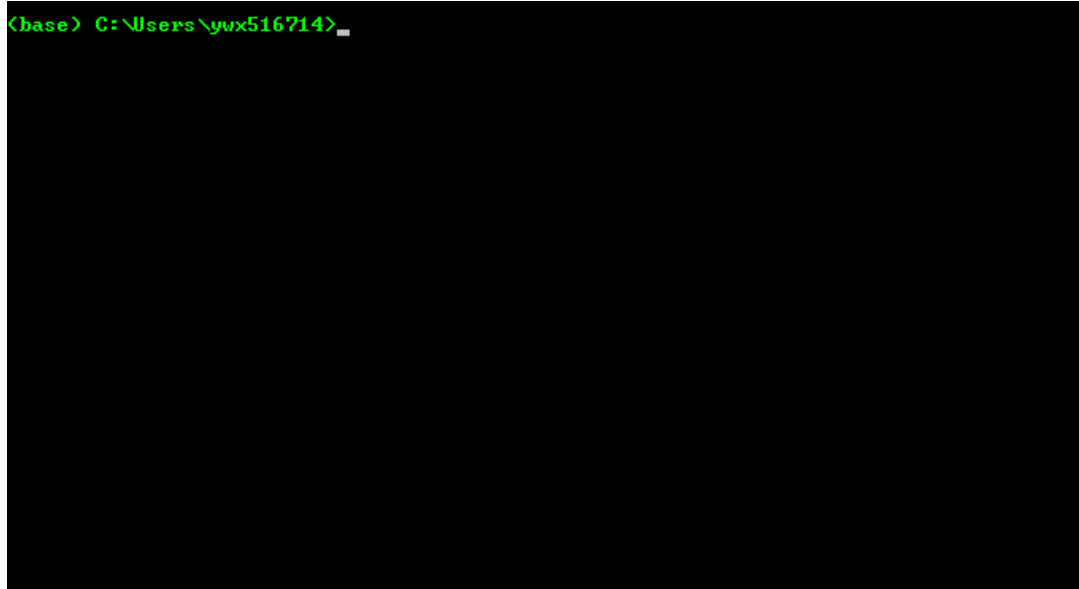


Figure 1-2 Anaconda Prompt

Install the wave module. Enter **pip install wave**. The result is as follows:

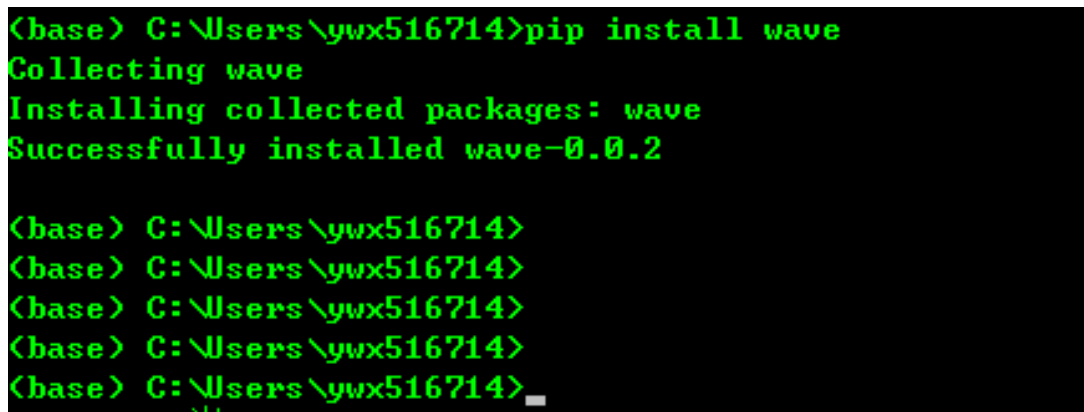


Figure 1-3 install wave

Install other required Python frameworks by following the similar steps.

1.5 Procedure

This experiment is performed based on the wave framework. Main steps include:

- View audio data attributes.

- View the audio spectrum.
- View the audio waveform.

Step 1 Import related modules.

Code:

```
import wave as we
import matplotlib.pyplot as plt
import numpy as np
from scipy.io import wavfile
import matplotlib.pyplot as plt
from matplotlib.backend_bases import RendererBase
from scipy import signal
from scipy.io import wavfile
import os
from PIL import Image
from scipy.fftpack import fft
import warnings
warnings.filterwarnings("ignore")
```

Step 2 View basic attributes of the wav file.

Code:

```
filename = r'\HCIP-
AI_speech_process\ASR_EN_MOD\ASR\data\thchs30\train\A2_0.wav'
# Output information (sound channel, sampling width, frame rate, number of
frames, unique ID, and lossless information)
WAVE = we.open(filename)

for item in enumerate(WAVE.getparams()):
    print (item)
a = WAVE.getparams().nframes      # Total number of frames
print(a)
f = WAVE.getparams().framerate    # Sampling frequency
print(f)
sample_time = 1/f                # Interval of sampling points
time = a/f                       # Sound signal length
sample_frequency, audio_sequence = wavfile.read(filename)
print (audio_sequence,len(audio_sequence ))    # Size of each frame of a
sound signal
x_seq = np.arange(0,time,sample_time)
print(x_seq,len(x_seq))
```

Result:

```
(0, 2)
(1, 2)
(2, 48000)
(3, 350208)
(4, 'NONE')
(5, 'not compressed')
```

```

350208
48000
[[ 0 0]
 [ 0 0]
 [ 0 0]
 ...
 [-187 -187]
 [-186 -186]
 [-181 -181]] 350208
[0.00000000e+00 2.08333333e-05 4.16666667e-05 ... 7.29593750e+00
 7.29595833e+00 7.29597917e+00] 350208

```

Step 3 View the waveform sequence of the wav file.

Code:

```

plt.plot(x_seq, audio_sequence, 'blue' )
plt.xlabel('time (s)')
plt.show()

```

Result:

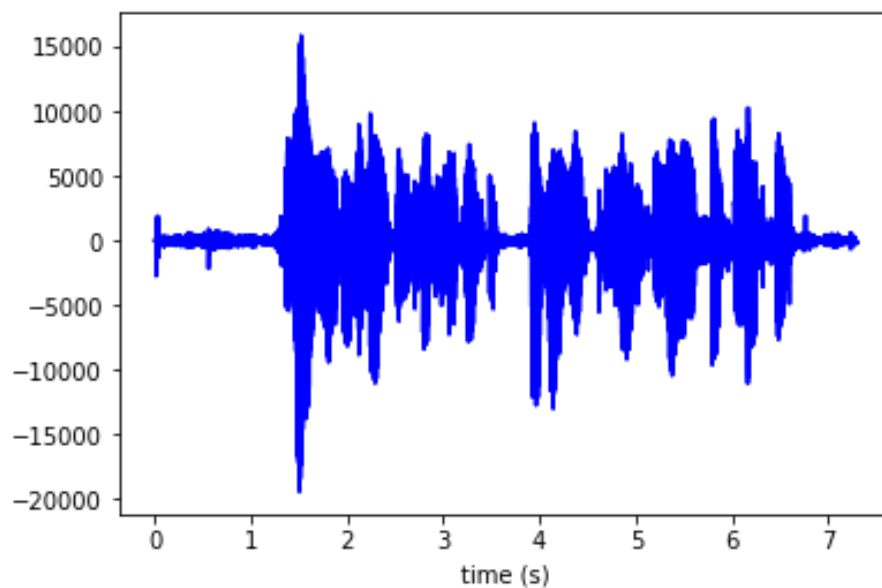


Figure 1-4 Result

Step 4 Obtain the file path.

Code:

```

audio_path = r'\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\train\audio'
pict_Path = r'\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\pics\train'

samples = []

```

```
# Check whether the file exists. If no file exists here, create a file.
if not os.path.exists(pict_Path):
    os.makedirs(pict_Path)

subFolderList = []
for x in os.listdir(audio_path):
    if os.path.isdir(audio_path + '/' + x):
        subFolderList.append(x)
    if not os.path.exists(pict_Path + '/' + x):
        os.makedirs(pict_Path + '/' + x)

# View the sub-file names and number of sub-files.
print("----list----:", subFolderList)
print("----len----:", len(subFolderList))
```

Result:

```
----list----: ['eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'ten',
'three', 'two']
----len----: 10
```

Step 5 Collect statistics of speech in each sub-folder.**Code:**

```
# count
sample_audio = []
total = 0
for x in subFolderList:

    # Obtain all wav files.
    all_files = [y for y in os.listdir(audio_path+'/' + x) if '.wav' in y]
    total += len(all_files)
    sample_audio.append(audio_path+'/' + x + '/' + all_files[0])

    # View the number of files in each sub-folder.
    print('%s : count:%d ' % (x , len(all_files)))

# View the total number of wav files.
print("TOTAL:", total)
```

Result:

```
eight : count:9
five : count:9
four : count:9
nine : count:9
one : count:9
seven : count:9
six : count:9
ten : count:9
three : count:9
two : count:9
TOTAL: 90
```

Step 6 View the first file in each sub-folder.**Code:**

```
for x in sample_audio:
    print(x)
```

Result:

```
\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\train\audio\eight/eight_1.wav
\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\train\audio/five/five_1.wav
\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\train\audio/four/four_1.wav
\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\train\audio/nine/nine_1.wav
\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\train\audio/one/one_1.wav
\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\train\audio/seven/seven_1.wav
\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\train\audio/six/six_1.wav
\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\train\audio/ten/ten_1.wav
\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\train\audio/three/three_1.wav
\HCIP-AI_speech_process\ASR_EN_MOD\ASR\data\train\audio/two/two_1.wav
```

Step 7 Create a spectrum processing function.**Code:**

```
def log_spectrogram(audio, sample_rate, window_size=20,
                    step_size=10, eps=1e-10):
    nperseg = int(round(window_size * sample_rate / 1e3))
    noverlap = int(round(step_size * sample_rate / 1e3))
    freqs, _, spec = signal.spectrogram(audio,
                                         fs=sample_rate,
                                         window='hann',
                                         nperseg=nperseg,
                                         noverlap=noverlap,
                                         detrend=False)
    return freqs, np.log(spec.T.astype(np.float32) + eps)
```

Step 8 Visualize one spectrum of multiple samples.**Code:**

```
fig = plt.figure(figsize=(20,20))

for i, filepath in enumerate(sample_audio[:16]):
    # Make subplots
    plt.subplot(4,4,i+1)

    # pull the labels
    label = filepath.split('/')[-2]
    plt.title(label)

    # create spectrogram
    samplerate, test_sound = wavfile.read(filepath)
```

```
_, spectrogram = log_spectrogram(test_sound, samplerate)

plt.imshow(spectrogram.T, aspect='auto', origin='lower')
plt.axis('off')
plt.show()
```

Result:

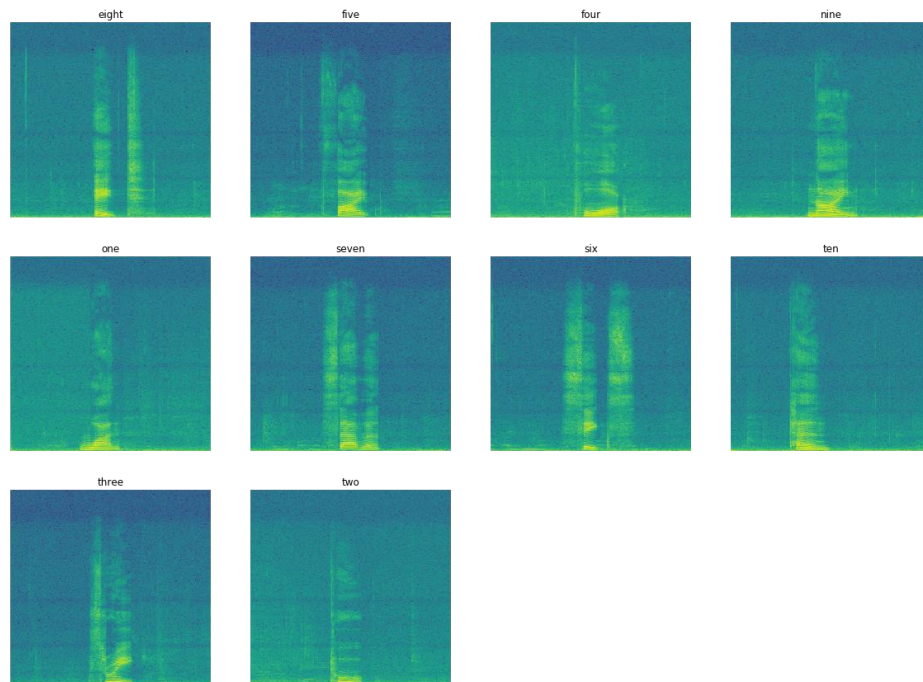


Figure 1-5 Result

Step 9 Visualize multiple spectrums of one sample.

Code:

```
one_samples = [audio_path + '/' + 'one/' + y for y in os.listdir(audio_path + '/' + 'one/')[1:9]]

fig = plt.figure(figsize=(10,10))

for i, filepath in enumerate(one_samples):
    # Make subplots
    plt.subplot(3,3,i+1)

    # pull the labels
    label = filepath.split('/')[-1]
    plt.title('"one": ' + label)

    # create spectrogram
    samplerate, test_sound = wavfile.read(filepath)
    _, spectrogram = log_spectrogram(test_sound, samplerate)
```

```
plt.imshow(spectrogram.T, aspect='auto', origin='lower')
plt.axis('off')
plt.show()
```

Result:

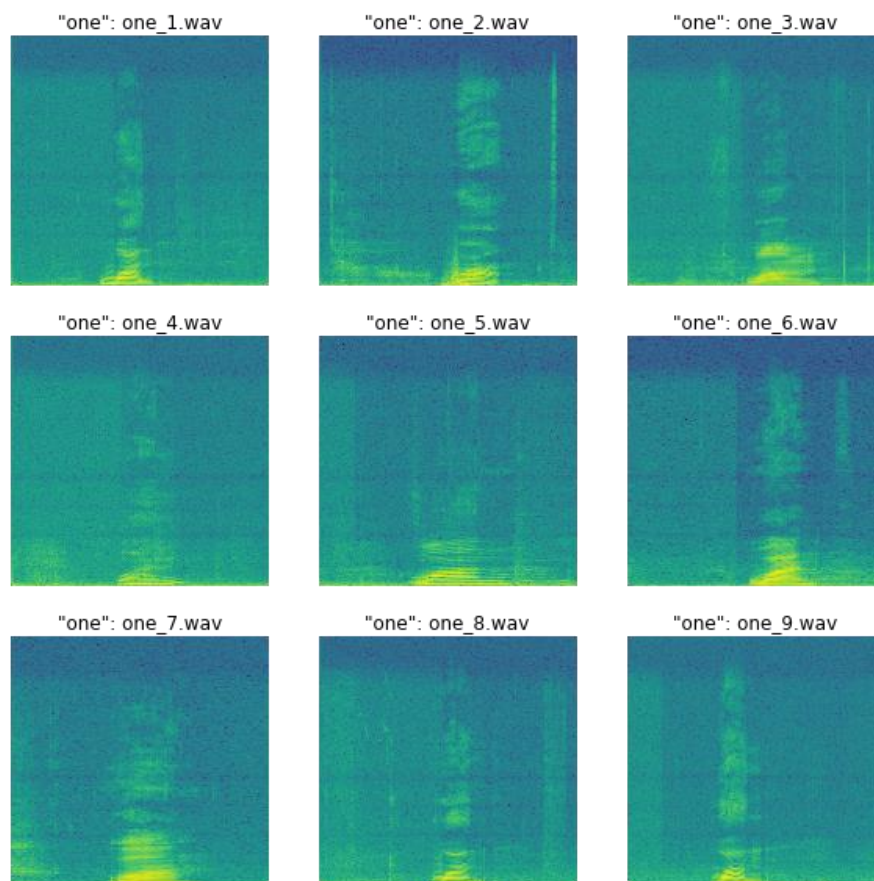


Figure 1-6 Result

Step 10 Visualize the waveforms of multiple samples.

Code:

```
fig = plt.figure(figsize=(10,10))
for i, filepath in enumerate(sample_audio[:16]):
    plt.subplot(4,4,i+1)
    samplerate, test_sound = wavfile.read(filepath)
    plt.title(filepath.split('/')[-2])
    plt.axis('off')
    plt.plot(test_sound)
plt.show()
```

Result:

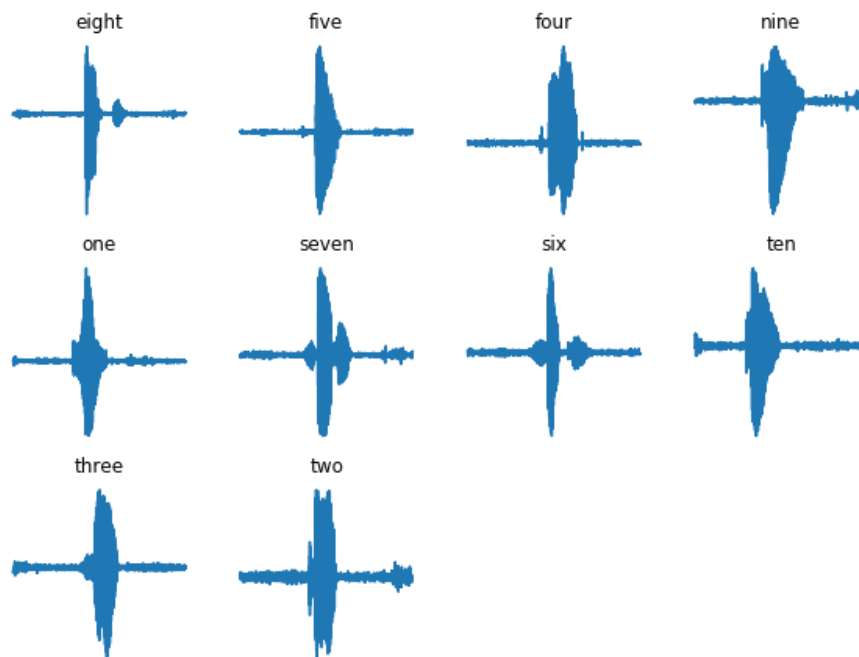


Figure 1-7 Result

Step 11 Visualize multiple waveforms of one sample.

Code:

```
fig = plt.figure(figsize=(8,8))
for i, filepath in enumerate(yes_samples):
    plt.subplot(3,3,i+1)
    samplerate, test_sound = wavfile.read(filepath)
    plt.title(filepath.split('/')[-2])
    plt.axis('off')
    plt.plot(test_sound)
plt.show()
```

Result:

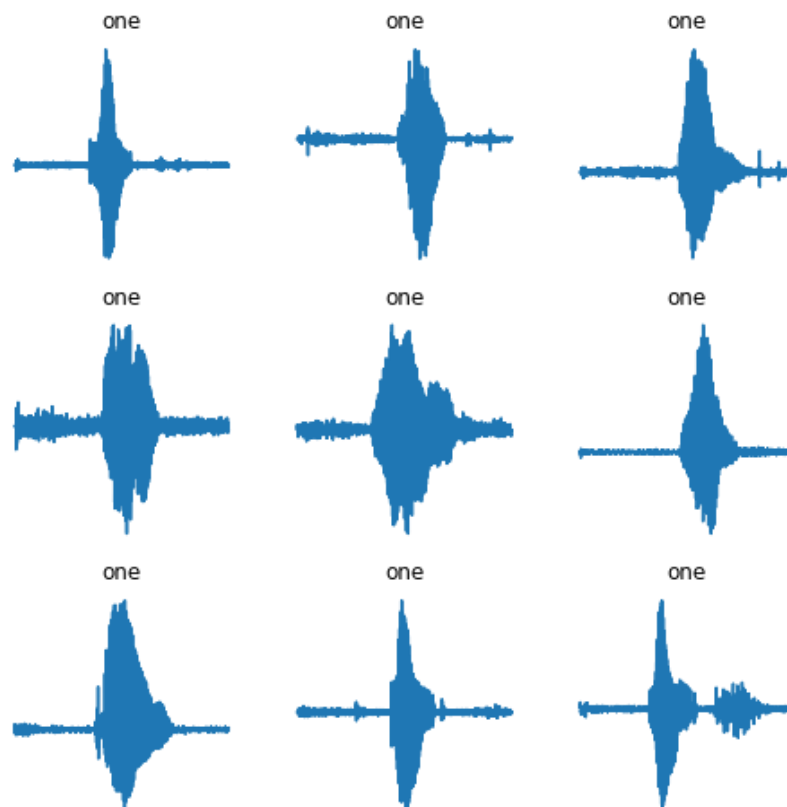


Figure 1-8 Result

----End

1.6 Summary

This experiment is a speech data pre-processing experiment based on the Python language, wave speech processing framework, and open source data set. It mainly includes viewing of basic speech data and processing of waveform and spectrum files. Visualization and display of specific values help trainees view essential attributes of speech data more clearly.

2 TTS

2.1 Introduction to the Experiment

TTS is the technology used to convert text into speech. The traditional methods that used in TTS can be roughly divided into two branches, the first branch is the parameters synthesis and the second branch is the wave concatenate. Parameters synthesis is hard to operation because the proper parameters are hard to adjust. And the wave concatenate has wider applications than parameters synthesis because it is easy to apply. But as the blossom of deep neural network, these models proved good performance on TTS tasks. So the most widely used method in modern times is the deep neural network model.

In this experiment the model we use to tackle these tasks is the deep neural network model, namely Seq2Seq with attention. Of course, there are many other models we can use, we choose this model because it has been proved have good performance and easy for new learner to understand. This model is often used in many other application, such as machine translation and text generation.

Tensorflow is the framework used in this experiment to build the deep neural network and to process the training of this model.

2.2 Purpose

This experiment introduces the detailed step of the whole TTS task, such as how to preprocess the raw data, how to label the raw data and how to build and train model in tensorflow. Because the amount of code is large and the modules are embedded into each other, especially the training process will take a lot of time, so our main purpose of this experiment is to help new learners to understand the whole framework and process of TTS implemented by tensorflow and after this experiment learners can build their own models to tackle specific TTS applications.

2.3 Knowledge Required

This experiment requires knowledge are as follows:

- Syntactical basis of the Python language and hands-on operation capability
- Understanding the use of tensorflow
- Have basic knowledge of the pipeline of text to speech
- Understand basic concepts and process of machine learning
- Understand basic knowledge of deep neural network

2.4 Procedure

For the readability and compact style of code, we divide the whole process of this project into two steps. In first step, we build the modules that will be used in many places. In second step, we build the whole train process based on the modules we have built in first step.

2.4.1 Build Basic Modules

Step 1 **Build the utils module.**

```
import numpy as np
import librosa
import os, copy
import matplotlib
matplotlib.use('pdf')
import matplotlib.pyplot as plt
from scipy import signal

from hyperparams import Hyperparams as hp
import tensorflow as tf

def get_spectrograms(fpath):
    '''Parse the wave file in `fpath` and
    Returns normalized melspectrogram and linear spectrogram.

    Args:
        fpath: A string. The full path of a sound file.

    Returns:
        mel: A 2d array of shape (T, n_mels) and dtype of float32.
        mag: A 2d array of shape (T, 1+n_fft/2) and dtype of float32.
    '''
    # Loading sound file
    y, sr = librosa.load(fpath, sr=hp.sr)
```

```
# Trimming
y, _ = librosa.effects.trim(y)

# Preemphasis
y = np.append(y[0], y[1:] - hp.preemphasis * y[:-1])

# stft
linear = librosa.stft(y=y,
                      n_fft=hp.n_fft,
                      hop_length=hp.hop_length,
                      win_length=hp.win_length)

# magnitude spectrogram
mag = np.abs(linear) # (1+n_fft//2, T)

# mel spectrogram
mel_basis = librosa.filters.mel(hp.sr, hp.n_fft, hp.n_mels) # (n_mels,
1+n_fft//2)
mel = np.dot(mel_basis, mag) # (n_mels, t)

# to decibel
mel = 20 * np.log10(np.maximum(1e-5, mel))
mag = 20 * np.log10(np.maximum(1e-5, mag))

# normalize
mel = np.clip((mel - hp.ref_db + hp.max_db) / hp.max_db, 1e-8, 1)
mag = np.clip((mag - hp.ref_db + hp.max_db) / hp.max_db, 1e-8, 1)

# Transpose
mel = mel.T.astype(np.float32) # (T, n_mels)
mag = mag.T.astype(np.float32) # (T, 1+n_fft//2)

return mel, mag

def spectrogram2wav(mag):
    '''# Generate wave file from linear magnitude spectrogram

    Args:
        mag: A numpy array of (T, 1+n_fft//2)

    Returns:
        wav: A 1-D numpy array.
    '''
    # transpose
    mag = mag.T

    # de-normalize
    mag = (np.clip(mag, 0, 1) * hp.max_db) - hp.max_db + hp.ref_db

    # to amplitude
    mag = np.power(10.0, mag * 0.05)

    # wav reconstruction
    wav = griffin_lim(mag**hp.power)
```

```
# de-preemphasis
wav = signal.lfilter([1], [1, -hp.preemphasis], wav)

# trim
wav, _ = librosa.effects.trim(wav)

return wav.astype(np.float32)

def griffin_lim(spectrogram):
    '''Applies Griffin-Lim's raw.'''
    X_best = copy.deepcopy(spectrogram)
    for i in range(hp.n_iter):
        X_t = invert_spectrogram(X_best)
        est = librosa.stft(X_t, hp.n_fft, hp.hop_length,
win_length=hp.win_length)
        phase = est / np.maximum(1e-8, np.abs(est))
        X_best = spectrogram * phase
        X_t = invert_spectrogram(X_best)
        y = np.real(X_t)

    return y

def invert_spectrogram(spectrogram):
    '''Applies inverse fft.
    Args:
        spectrogram: [1+n_fft//2, t]
    '''
    return librosa.istft(spectrogram, hp.hop_length, win_length=hp.win_length,
window="hann")

def plot_alignment(alignment, gs, dir=hp.logdir):
    """Plots the alignment.

    Args:
        alignment: A numpy array with shape of (encoder_steps, decoder_steps)
        gs: (int) global step.
        dir: Output path.
    """
    if not os.path.exists(dir): os.mkdir(dir)

    fig, ax = plt.subplots()
    im = ax.imshow(alignment)

    fig.colorbar(im)
    plt.title('{} Steps'.format(gs))
    plt.savefig('{}alignment_{}.png'.format(dir, gs), format='png')
    plt.close(fig)

def guided_attention(g=0.2):
    '''Guided attention. Refer to page 3 on the paper.'''
    W = np.zeros((hp.max_N, hp.max_T), dtype=np.float32)
    for n_pos in range(W.shape[0]):
        for t_pos in range(W.shape[1]):
```

```

        W[n_pos, t_pos] = 1 - np.exp(-(t_pos / float(hp.max_T) - n_pos /
float(hp.max_N)) ** 2 / (2 * g * g))
    return W

def learning_rate_decay(init_lr, global_step, warmup_steps = 4000.0):
    '''Noam scheme from tensor2tensor'''
    step = tf.to_float(global_step + 1)
    return init_lr * warmup_steps**0.5 * tf.minimum(step * warmup_steps**-1.5,
step**-0.5)

def load_spectrograms(fpath):
    '''Read the wave file in `fpath`
and extracts spectrograms'''

    fname = os.path.basename(fpath)
    mel, mag = get_spectrograms(fpath)
    t = mel.shape[0]

    # Marginal padding for reduction shape sync.
    num_paddings = hp.r - (t % hp.r) if t % hp.r != 0 else 0
    mel = np.pad(mel, [[0, num_paddings], [0, 0]], mode="constant")
    mag = np.pad(mag, [[0, num_paddings], [0, 0]], mode="constant")

    # Reduction
    mel = mel[:, :hp.r, :]
    return fname, mel, mag

```

Step 2 Bulid the basic modules for network.

```

import tensorflow as tf

def embed(inputs, vocab_size, num_units, zero_pad=True, scope="embedding",
reuse=None):
    '''Embeds a given tensor.

    Args:
        inputs: A `Tensor` with type `int32` or `int64` containing the ids
            to be looked up in `lookup table`.
        vocab_size: An int. Vocabulary size.
        num_units: An int. Number of embedding hidden units.
        zero_pad: A boolean. If True, all the values of the first row (id 0)
            should be constant zeros.
        scope: Optional scope for `variable_scope`.
        reuse: Boolean, whether to reuse the weights of a previous layer
            by the same name.

    Returns:
        A `Tensor` with one more rank than inputs's. The last dimensionality
            should be `num_units`.
    '''
    with tf.variable_scope(scope, reuse=reuse):
        lookup_table = tf.get_variable('lookup_table',
                                       dtype=tf.float32,
                                       shape=[vocab_size, num_units],

```

```
initializer=tf.truncated_normal_initializer(mean=0.0, stddev=0.1))
    if zero_pad:
        lookup_table = tf.concat((tf.zeros(shape=[1, num_units]),
                                      lookup_table[1:, :]), 0)

    outputs = tf.nn.embedding_lookup(lookup_table, inputs)

    return outputs

def normalize(inputs,
              scope="normalize",
              reuse=None):
    '''Applies layer normalization that normalizes along the last axis.

    Args:
        inputs: A tensor with 2 or more dimensions, where the first dimension
        has
            `batch_size`. The normalization is over the last dimension.
        scope: Optional scope for `variable_scope`.
        reuse: Boolean, whether to reuse the weights of a previous layer
            by the same name.

    Returns:
        A tensor with the same shape and data dtype as `inputs`.
    '''
    outputs = tf.contrib.layers.layer_norm(inputs,
                                           begin_norm_axis=-1,
                                           scope=scope,
                                           reuse=reuse)

    return outputs

def highwaynet(inputs, num_units=None, scope="highwaynet", reuse=None):
    '''Highway networks, see https://arxiv.org/abs/1505.00387

    Args:
        inputs: A 3D tensor of shape [N, T, W].
        num_units: An int or `None`. Specifies the number of units in the
        highway layer
            or uses the input size if `None`.
        scope: Optional scope for `variable_scope`.
        reuse: Boolean, whether to reuse the weights of a previous layer
            by the same name.

    Returns:
        A 3D tensor of shape [N, T, W].
    '''
    if not num_units:
        num_units = inputs.get_shape()[-1]

    with tf.variable_scope(scope, reuse=reuse):
        H = tf.layers.dense(inputs, units=num_units, activation=tf.nn.relu,
                             name="dense1")
```

```
T = tf.layers.dense(inputs, units=num_units, activation=tf.nn.sigmoid,
                    bias_initializer=tf.constant_initializer(-1.0),
name="dense2")
    outputs = H * T + inputs * (1. - T)
    return outputs

def conv1d(inputs,
          filters=None,
          size=1,
          rate=1,
          padding="SAME",
          dropout_rate=0,
          use_bias=True,
          activation_fn=None,
          training=True,
          scope="conv1d",
          reuse=None):
    '''
    Args:
        inputs: A 3-D tensor with shape of [batch, time, depth].
        filters: An int. Number of outputs (=activation maps)
        size: An int. Filter size.
        rate: An int. Dilation rate.
        padding: Either `same` or `valid` or `causal` (case-insensitive).
        dropout_rate: A float of [0, 1].
        use_bias: A boolean.
        activation_fn: A string.
        training: A boolean. If True, dropout is applied.
        scope: Optional scope for `variable_scope`.
        reuse: Boolean, whether to reuse the weights of a previous layer
            by the same name.

    Returns:
        A masked tensor of the same shape and dtypes as `inputs`.
    '''
    with tf.variable_scope(scope):
        if padding.lower() == "causal":
            # pre-padding for causality
            pad_len = (size - 1) * rate # padding size
            inputs = tf.pad(inputs, [[0, 0], [pad_len, 0], [0, 0]])
            padding = "valid"

        if filters is None:
            filters = inputs.get_shape().as_list()[-1]

        params = {"inputs": inputs, "filters": filters, "kernel_size": size,
                  "dilation_rate": rate, "padding": padding, "use_bias":
use_bias,
                  "kernel_initializer":
tf.contrib.layers.variance_scaling_initializer(), "reuse": reuse}

        tensor = tf.layers.conv1d(**params)
        tensor = normalize(tensor)
        if activation_fn is not None:
            tensor = activation_fn(tensor)
```

```
        tensor = tf.layers.dropout(tensor, rate=dropout_rate,
training=training)

    return tensor

def hc(inputs,
        filters=None,
        size=1,
        rate=1,
        padding="SAME",
        dropout_rate=0,
        use_bias=True,
        activation_fn=None,
        training=True,
        scope="hc",
        reuse=None):
    '''
    Args:
        inputs: A 3-D tensor with shape of [batch, time, depth].
        filters: An int. Number of outputs (=activation maps)
        size: An int. Filter size.
        rate: An int. Dilation rate.
        padding: Either `same` or `valid` or `causal` (case-insensitive).
        use_bias: A boolean.
        activation_fn: A string.
        training: A boolean. If True, dropout is applied.
        scope: Optional scope for `variable_scope`.
        reuse: Boolean, whether to reuse the weights of a previous layer
            by the same name.

    Returns:
        A masked tensor of the same shape and dtypes as `inputs`.
    '''
    _inputs = inputs
    with tf.variable_scope(scope):
        if padding.lower() == "causal":
            # pre-padding for causality
            pad_len = (size - 1) * rate # padding size
            inputs = tf.pad(inputs, [[0, 0], [pad_len, 0], [0, 0]])
            padding = "valid"

        if filters is None:
            filters = inputs.get_shape().as_list()[-1]

        params = {"inputs": inputs, "filters": 2*filters, "kernel_size": size,
                  "dilation_rate": rate, "padding": padding, "use_bias":
use_bias,
                  "kernel_initializer":
tf.contrib.layers.variance_scaling_initializer(), "reuse": reuse}

        tensor = tf.layers.conv1d(**params)
        H1, H2 = tf.split(tensor, 2, axis=-1)
        H1 = normalize(H1, scope="H1")
```

```
H2 = normalize(H2, scope="H2")
H1 = tf.nn.sigmoid(H1, "gate")
H2 = activation_fn(H2, "info") if activation_fn is not None else H2
tensor = H1*H2 + (1.-H1)*_inputs

tensor = tf.layers.dropout(tensor, rate=dropout_rate,
training=training)

return tensor

def conv1d_transpose(inputs,
                    filters=None,
                    size=3,
                    stride=2,
                    padding='same',
                    dropout_rate=0,
                    use_bias=True,
                    activation=None,
                    training=True,
                    scope="conv1d_transpose",
                    reuse=None):
    """
    Args:
        inputs: A 3-D tensor with shape of [batch, time, depth].
        filters: An int. Number of outputs (=activation maps)
        size: An int. Filter size.
        rate: An int. Dilation rate.
        padding: Either `same` or `valid` or `causal` (case-insensitive).
        dropout_rate: A float of [0, 1].
        use_bias: A boolean.
        activation_fn: A string.
        training: A boolean. If True, dropout is applied.
        scope: Optional scope for `variable_scope`.
        reuse: Boolean, whether to reuse the weights of a previous layer
            by the same name.

    Returns:
        A tensor of the shape with [batch, time*2, depth].
    """
    with tf.variable_scope(scope, reuse=reuse):
        if filters is None:
            filters = inputs.get_shape().as_list()[-1]
        inputs = tf.expand_dims(inputs, 1)
        tensor = tf.layers.conv2d_transpose(inputs,
                                            filters=filters,
                                            kernel_size=(1, size),
                                            strides=(1, stride),
                                            padding=padding,
                                            activation=None,

kernel_initializer=tf.contrib.layers.variance_scaling_initializer(),
                        use_bias=use_bias)
        tensor = tf.squeeze(tensor, 1)
        tensor = normalize(tensor)
        if activation is not None:
```

```
        tensor = activation(tensor)

        tensor = tf.layers.dropout(tensor, rate=dropout_rate,
                                    training=training)

    return tensor
```

Step 3 Build the network.

```
from hyperparams import Hyperparams as hp
from modules import *
import tensorflow as tf

def TextEnc(L, training=True):
    '''
    Args:
        L: Text inputs. (B, N)

    Return:
        K: Keys. (B, N, d)
        V: Values. (B, N, d)
    '''
    i = 1
    tensor = embed(L,
                   vocab_size=len(hp.vocab),
                   num_units=hp.e,
                   scope="embed_{}".format(i)); i += 1
    tensor = conv1d(tensor,
                    filters=2*hp.d,
                    size=1,
                    rate=1,
                    dropout_rate=hp.dropout_rate,
                    activation_fn=tf.nn.relu,
                    training=training,
                    scope="C_{}".format(i)); i += 1
    tensor = conv1d(tensor,
                    size=1,
                    rate=1,
                    dropout_rate=hp.dropout_rate,
                    training=training,
                    scope="C_{}".format(i)); i += 1

    for _ in range(2):
        for j in range(4):
            tensor = hc(tensor,
                        size=3,
                        rate=3**j,
                        dropout_rate=hp.dropout_rate,
                        activation_fn=None,
                        training=training,
                        scope="HC_{}".format(i)); i += 1
    for _ in range(2):
        tensor = hc(tensor,
                    size=3,
                    rate=1,
```

```
        dropout_rate=hp.dropout_rate,
        activation_fn=None,
        training=training,
        scope="HC_{}".format(i)); i += 1

for _ in range(2):
    tensor = hc(tensor,
                size=1,
                rate=1,
                dropout_rate=hp.dropout_rate,
                activation_fn=None,
                training=training,
                scope="HC_{}".format(i)); i += 1

K, V = tf.split(tensor, 2, -1)
return K, V

def AudioEnc(S, training=True):
    '''
    Args:
        S: melspectrogram. (B, T/r, n_mels)

    Returns
        Q: Queries. (B, T/r, d)
    '''
    i = 1
    tensor = conv1d(S,
                    filters=hp.d,
                    size=1,
                    rate=1,
                    padding="CAUSAL",
                    dropout_rate=hp.dropout_rate,
                    activation_fn=tf.nn.relu,
                    training=training,
                    scope="C_{}".format(i)); i += 1
    tensor = conv1d(tensor,
                    size=1,
                    rate=1,
                    padding="CAUSAL",
                    dropout_rate=hp.dropout_rate,
                    activation_fn=tf.nn.relu,
                    training=training,
                    scope="C_{}".format(i)); i += 1
    tensor = conv1d(tensor,
                    size=1,
                    rate=1,
                    padding="CAUSAL",
                    dropout_rate=hp.dropout_rate,
                    training=training,
                    scope="C_{}".format(i)); i += 1
    for _ in range(2):
        for j in range(4):
            tensor = hc(tensor,
                        size=3,
                        rate=3**j,
```

```

        padding="CAUSAL",
        dropout_rate=hp.dropout_rate,
        training=training,
        scope="HC_{}".format(i)); i += 1
for _ in range(2):
    tensor = hc(tensor,
                size=3,
                rate=3,
                padding="CAUSAL",
                dropout_rate=hp.dropout_rate,
                training=training,
                scope="HC_{}".format(i)); i += 1

return tensor

def Attention(Q, K, V, mononotic_attention=False, prev_max_attentions=None):
    '''
    Args:
        Q: Queries. (B, T/r, d)
        K: Keys. (B, N, d)
        V: Values. (B, N, d)
        mononotic_attention: A boolean. At training, it is False.
        prev_max_attentions: (B,). At training, it is set to None.

    Returns:
        R: [Context Vectors; Q]. (B, T/r, 2d)
        alignments: (B, N, T/r)
        max_attentions: (B, T/r)
    '''
    A = tf.matmul(Q, K, transpose_b=True) * tf.rsqrt(tf.to_float(hp.d))
    if mononotic_attention: # for inference
        key_masks = tf.sequence_mask(prev_max_attentions, hp.max_N)
        reverse_masks = tf.sequence_mask(hp.max_N - hp.attention_win_size -
prev_max_attentions, hp.max_N)[: , ::-1]
        masks = tf.logical_or(key_masks, reverse_masks)
        masks = tf.tile(tf.expand_dims(masks, 1), [1, hp.max_T, 1])
        paddings = tf.ones_like(A) * (-2 ** 32 + 1) # (B, T/r, N)
        A = tf.where(tf.equal(masks, False), A, paddings)
    A = tf.nn.softmax(A) # (B, T/r, N)
    max_attentions = tf.argmax(A, -1) # (B, T/r)
    R = tf.matmul(A, V)
    R = tf.concat((R, Q), -1)

    alignments = tf.transpose(A, [0, 2, 1]) # (B, N, T/r)

    return R, alignments, max_attentions

def AudioDec(R, training=True):
    '''
    Args:
        R: [Context Vectors; Q]. (B, T/r, 2d)

    Returns:
        Y: Melspectrogram predictions. (B, T/r, n_mels)
    '''

```

```
i = 1
tensor = conv1d(R,
                filters=hp.d,
                size=1,
                rate=1,
                padding="CAUSAL",
                dropout_rate=hp.dropout_rate,
                training=training,
                scope="C_{}".format(i)); i += 1
for j in range(4):
    tensor = hc(tensor,
                size=3,
                rate=3**j,
                padding="CAUSAL",
                dropout_rate=hp.dropout_rate,
                training=training,
                scope="HC_{}".format(i)); i += 1

for _ in range(2):
    tensor = hc(tensor,
                size=3,
                rate=1,
                padding="CAUSAL",
                dropout_rate=hp.dropout_rate,
                training=training,
                scope="HC_{}".format(i)); i += 1
for _ in range(3):
    tensor = conv1d(tensor,
                    size=1,
                    rate=1,
                    padding="CAUSAL",
                    dropout_rate=hp.dropout_rate,
                    activation_fn=tf.nn.relu,
                    training=training,
                    scope="C_{}".format(i)); i += 1

# mel_hats
logits = conv1d(tensor,
                filters=hp.n_mels,
                size=1,
                rate=1,
                padding="CAUSAL",
                dropout_rate=hp.dropout_rate,
                training=training,
                scope="C_{}".format(i)); i += 1
Y = tf.nn.sigmoid(logits) # mel_hats

return logits, Y

def SSRN(Y, training=True):
    '''
    Args:
        Y: Melspectrogram Predictions. (B, T/r, n_mels)

    Returns:
```

```

Z: Spectrogram Predictions. (B, T, 1+n_fft/2)
'''

i = 1 # number of layers

# -> (B, T/r, c)
tensor = conv1d(Y,
                 filters=hp.c,
                 size=1,
                 rate=1,
                 dropout_rate=hp.dropout_rate,
                 training=training,
                 scope="C_{}".format(i)); i += 1
for j in range(2):
    tensor = hc(tensor,
                 size=3,
                 rate=3**j,
                 dropout_rate=hp.dropout_rate,
                 training=training,
                 scope="HC_{}".format(i)); i += 1
for _ in range(2):
    # -> (B, T/2, c) -> (B, T, c)
    tensor = conv1d_transpose(tensor,
                               scope="D_{}".format(i),
                               dropout_rate=hp.dropout_rate,
                               training=training,); i += 1
    for j in range(2):
        tensor = hc(tensor,
                     size=3,
                     rate=3**j,
                     dropout_rate=hp.dropout_rate,
                     training=training,
                     scope="HC_{}".format(i)); i += 1
# -> (B, T, 2*c)
tensor = conv1d(tensor,
                 filters=2*hp.c,
                 size=1,
                 rate=1,
                 dropout_rate=hp.dropout_rate,
                 training=training,
                 scope="C_{}".format(i)); i += 1
for _ in range(2):
    tensor = hc(tensor,
                 size=3,
                 rate=1,
                 dropout_rate=hp.dropout_rate,
                 training=training,
                 scope="HC_{}".format(i)); i += 1
# -> (B, T, 1+n_fft/2)
tensor = conv1d(tensor,
                 filters=1+hp.n_fft//2,
                 size=1,
                 rate=1,
                 dropout_rate=hp.dropout_rate,
                 training=training,

```

```

        scope="C_{}".format(i)); i += 1

for _ in range(2):
    tensor = conv1d(tensor,
                    size=1,
                    rate=1,
                    dropout_rate=hp.dropout_rate,
                    activation_fn=tf.nn.relu,
                    training=training,
                    scope="C_{}".format(i)); i += 1
logits = conv1d(tensor,
                size=1,
                rate=1,
                dropout_rate=hp.dropout_rate,
                training=training,
                scope="C_{}".format(i))
Z = tf.nn.sigmoid(logits)
return logits, Z

```

2.4.2 Build The Whole Training Process

Step 1 Build hyperparameters.

```

class Hyperparams:
    '''Hyper parameters'''
    # pipeline
    prepro = False # if True, run `python prepro.py` first before running
                    `python train.py`.

    # signal processing
    sr = 22050 # Sampling rate.
    n_fft = 2048 # fft points (samples)
    frame_shift = 0.0125 # seconds
    frame_length = 0.05 # seconds
    hop_length = int(sr * frame_shift) # samples. =276.
    win_length = int(sr * frame_length) # samples. =1102.
    n_mels = 80 # Number of Mel banks to generate
    power = 1.5 # Exponent for amplifying the predicted magnitude
    n_iter = 50 # Number of inversion iterations
    preemphasis = .97
    max_db = 100
    ref_db = 20

    # Model
    r = 4 # Reduction factor. Do not change this.
    dropout_rate = 0.05
    e = 128 # == embedding
    d = 256 # == hidden units of Text2Mel
    c = 512 # == hidden units of SSRN
    attention_win_size = 3

    # data
    data = "LJSpeech-1.1"
    # data = "/data/private/voice/kate"

```

```
test_data = 'harvard_sentences.txt'
vocab = "PE abcdefghijklmnopqrstuvwxyz'.'" # P: Padding, E: EOS.
max_N = 180 # Maximum number of characters.
max_T = 210 # Maximum number of mel frames.

# training scheme
lr = 0.001 # Initial learning rate.
logdir = "logdir/LJ01"
sampledir = 'samples'
B = 32 # batch size
num_iterations = 2000000
```

Step 2 Build the data load function

```
from hyperparams import Hyperparams as hp
import numpy as np
import tensorflow as tf
from utils import *
import codecs
import re
import os
import unicodedata

def load_vocab():
    char2idx = {char: idx for idx, char in enumerate(hp.vocab)}
    idx2char = {idx: char for idx, char in enumerate(hp.vocab)}
    return char2idx, idx2char

def text_normalize(text):
    text = ''.join(char for char in unicodedata.normalize('NFD', text)
                    if unicodedata.category(char) != 'Mn') # Strip accents

    text = text.lower()
    text = re.sub("[^{}]" .format(hp.vocab), " ", text)
    text = re.sub("[ ]+", " ", text)
    return text

def load_data(mode="train"):
    '''Loads data
    Args:
        mode: "train" or "synthesize".
    '''
    # Load vocabulary
    char2idx, idx2char = load_vocab()

    if mode=="train":
        if "LJ" in hp.data:
            # Parse
            fpaths, text_lengths, texts = [], [], []
            transcript = os.path.join(hp.data, 'metadata.csv')
            lines = codecs.open(transcript, 'r', 'utf-8').readlines()
            for line in lines:
                fname, _, text = line.strip().split("|")

                fpath = os.path.join(hp.data, "wavs", fname + ".wav")
```

```

        fpaths.append(fpath)

        text = text_normalize(text) + "E" # E: EOS
        text = [char2idx[char] for char in text]
        text_lengths.append(len(text))
        texts.append(np.array(text, np.int32).tostring())

    return fpaths, text_lengths, texts
else: # nick or kate
    # Parse
    fpaths, text_lengths, texts = [], [], []
    transcript = os.path.join(hp.data, 'metadata.csv')
    lines = codecs.open(transcript, 'r', 'utf-8').readlines()
    for line in lines:
        fname, _, text, is_inside_quotes, duration =
line.strip().split("|")
        duration = float(duration)
        if duration > 10. : continue

        fpath = os.path.join(hp.data, fname)
        fpaths.append(fpath)

        text += "E" # E: EOS
        text = [char2idx[char] for char in text]
        text_lengths.append(len(text))
        texts.append(np.array(text, np.int32).tostring())

    return fpaths, text_lengths, texts

else: # synthesize on unseen test text.
    # Parse
    lines = codecs.open(hp.test_data, 'r', 'utf-8').readlines()[1:]
    sents = [text_normalize(line.split(" ", 1)[-1].strip() + "E" for line
in lines) # text normalization, E: EOS
    texts = np.zeros((len(sents), hp.max_N), np.int32)
    for i, sent in enumerate(sents):
        texts[i, :len(sent)] = [char2idx[char] for char in sent]
    return texts

def get_batch():
    """Loads training data and put them in queues"""
    with tf.device('/cpu:0'):
        # Load data
        fpaths, text_lengths, texts = load_data() # list
        maxlen, minlen = max(text_lengths), min(text_lengths)

        # Calc total batch count
        num_batch = len(fpaths) // hp.B

        # Create Queues
        fpath, text_length, text = tf.train.slice_input_producer([fpaths,
text_lengths, texts], shuffle=True)

        # Parse
        text = tf.decode_raw(text, tf.int32) # (None,)

```

```

if hp.prepro:
    def _load_spectrograms(fpath):
        fname = os.path.basename(fpath)
        mel = "mels/{}".format(fname.replace("wav", "npz"))
        mag = "mags/{}".format(fname.replace("wav", "npz"))
        return fname, np.load(mel), np.load(mag)

    fname, mel, mag = tf.py_func(_load_spectrograms, [fpath],
[tf.string, tf.float32, tf.float32])
    else:
        fname, mel, mag = tf.py_func(load_spectrograms, [fpath],
[tf.string, tf.float32, tf.float32]) # (None, n_mels)

    # Add shape information
    fname.set_shape(())
    text.set_shape((None,))
    mel.set_shape((None, hp.n_mels))
    mag.set_shape((None, hp.n_fft//2+1))

    # Batching
    _, (texts, mels, mags, fnames) =
tf.contrib.training.bucket_by_sequence_length(
    input_length=text_length,
    tensors=[text, mel, mag, fname],
    batch_size=hp.B,
    bucket_boundaries=[i for i in
range(minlen + 1, maxlen - 1, 20)],
    num_threads=8,
    capacity=hp.B*4,
    dynamic_pad=True)

    return texts, mels, mags, fnames, num_batch

```

Step 3 Build folder.

```

from utils import load_spectrograms
import os
from data_load import load_data
import numpy as np
import tqdm

# Load data
fpaths, _, _ = load_data() # list

for fpath in tqdm.tqdm(fpaths):
    fname, mel, mag = load_spectrograms(fpath)
    if not os.path.exists("mels"): os.mkdir("mels")
    if not os.path.exists("mags"): os.mkdir("mags")

    np.save("mels/{}".format(fname.replace("wav", "npz")), mel)
    np.save("mags/{}".format(fname.replace("wav", "npz")), mag)

```

Step 4 Build train process.

```
# -*- coding: utf-8 -*-
# /usr/bin/python2
'''
By kyubyong park. kbpark.linguist@gmail.com.
https://www.github.com/kyubyong/dc_tts
'''

from tqdm import tqdm

from data_load import get_batch, load_vocab
from hyperparams import Hyperparams as hp
from modules import *
from networks import TextEnc, AudioEnc, AudioDec, Attention, SSRN
import tensorflow as tf
from utils import *
import sys

class Graph:
    def __init__(self, num=1, mode="train"):
        '''
        Args:
            num: Either 1 or 2. 1 for Text2Mel 2 for SSRN.
            mode: Either "train" or "synthesize".
        '''
        # Load vocabulary
        self.char2idx, self.idx2char = load_vocab()

        # Set flag
        training = True if mode=="train" else False

        # Graph
        # Data Feeding
        ## L: Text. (B, N), int32
        ## mels: Reduced melspectrogram. (B, T/r, n_mels) float32
        ## mags: Magnitude. (B, T, n_fft//2+1) float32
        if mode=="train":
            self.L, self.mels, self.mags, self.fnames, self.num_batch = \
get_batch()
            self.prev_max attentions = tf.ones(shape=(hp.B,), dtype=tf.int32)
            self.gts = tf.convert_to_tensor(guided_attention())
        else: # Synthesize
            self.L = tf.placeholder(tf.int32, shape=(None, None))
            self.mels = tf.placeholder(tf.float32, shape=(None, None,
hp.n_mels))
            self.prev_max attentions = tf.placeholder(tf.int32, shape=(None,))

        if num==1 or (not training):
            with tf.variable_scope("Text2Mel"):
                # Get S or decoder inputs. (B, T//r, n_mels)
                self.S = tf.concat((tf.zeros_like(self.mels[:, :1, :]),
self.mels[:, :-1, :]), 1)
```

```

# Networks
with tf.variable_scope("TextEnc"):
    self.K, self.V = TextEnc(self.L, training=training) # (N,
Tx, e)

    with tf.variable_scope("AudioEnc"):
        self.Q = AudioEnc(self.S, training=training)

    with tf.variable_scope("Attention"):
        # R: (B, T/r, 2d)
        # alignments: (B, N, T/r)
        # max_attentions: (B,)
        self.R, self.alignments, self.max_attentions =
Attention(self.Q, self.K, self.V,

monotonic_attention=(not training),

prev_max_attentions=self.prev_max_attentions)
    with tf.variable_scope("AudioDec"):
        self.Y_logits, self.Y = AudioDec(self.R, training=training) #
(B, T/r, n_mels)
    else: # num==2 & training. Note that during training,
        # the ground truth melspectrogram values are fed.
        with tf.variable_scope("SSRN"):
            self.Z_logits, self.Z = SSRN(self.mels, training=training)

    if not training:
        # During inference, the predicted melspectrogram values are fed.
        with tf.variable_scope("SSRN"):
            self.Z_logits, self.Z = SSRN(self.Y, training=training)

    with tf.variable_scope("gs"):
        self.global_step = tf.Variable(0, name='global_step',
trainable=False)

    if training:
        if num==1: # Text2Mel
            # mel L1 loss
            self.loss_mels = tf.reduce_mean(tf.abs(self.Y - self.mels))

            # mel binary divergence loss
            self.loss_bdl =
tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=self.Y_logits,
labels=self.mels))

            # guided_attention loss
            self.A = tf.pad(self.alignments, [(0, 0), (0, hp.max_N), (0,
hp.max_T)], mode="CONSTANT", constant_values=-1.)[:, :hp.max_N, :hp.max_T]
            self.attention_masks = tf.to_float(tf.not_equal(self.A, -1))
            self.loss_att = tf.reduce_sum(tf.abs(self.A * self.gts) *
self.attention_masks)
            self.mask_sum = tf.reduce_sum(self.attention_masks)
            self.loss_att /= self.mask_sum

            # total loss

```

```

        self.loss = self.loss_mels + self.loss_bd1 + self.loss_att

        tf.summary.scalar('train/loss_mels', self.loss_mels)
        tf.summary.scalar('train/loss_bd1', self.loss_bd1)
        tf.summary.scalar('train/loss_att', self.loss_att)
        tf.summary.image('train/mel_gt',
            tf.expand_dims(tf.transpose(self.mels[:1], [0, 2, 1]), -1))
        tf.summary.image('train/mel_hat',
            tf.expand_dims(tf.transpose(self.Y[:1], [0, 2, 1]), -1))
    else: # SSRN
        # mag L1 loss
        self.loss_mags = tf.reduce_mean(tf.abs(self.Z - self.mags))

        # mag binary divergence loss
        self.loss_bd2 =
            tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=self.Z_logits,
                labels=self.mags))

        # total loss
        self.loss = self.loss_mags + self.loss_bd2

        tf.summary.scalar('train/loss_mags', self.loss_mags)
        tf.summary.scalar('train/loss_bd2', self.loss_bd2)
        tf.summary.image('train/mag_gt',
            tf.expand_dims(tf.transpose(self.mags[:1], [0, 2, 1]), -1))
        tf.summary.image('train/mag_hat',
            tf.expand_dims(tf.transpose(self.Z[:1], [0, 2, 1]), -1))

    # Training Scheme
    self.lr = learning_rate_decay(hp.lr, self.global_step)
    self.optimizer = tf.train.AdamOptimizer(learning_rate=self.lr)
    tf.summary.scalar("lr", self.lr)

    ## gradient clipping
    self.gvs = self.optimizer.compute_gradients(self.loss)
    self.clipped = []
    for grad, var in self.gvs:
        grad = tf.clip_by_value(grad, -1., 1.)
        self.clipped.append((grad, var))
    self.train_op = self.optimizer.apply_gradients(self.clipped,
        global_step=self.global_step)

    # Summary
    self.merged = tf.summary.merge_all()

if __name__ == '__main__':
    # argument: 1 or 2. 1 for Text2mel, 2 for SSRN.
    num = int(sys.argv[1])

    g = Graph(num=num); print("Training Graph loaded")

    logdir = hp.logdir + "-" + str(num)
    sv = tf.train.Supervisor(logdir=logdir, save_model_secs=0,
        global_step=g.global_step)

```

```
with sv.managed_session() as sess:
    while 1:
        for _ in tqdm(list(range(g.num_batch)), total=g.num_batch,
ncols=70, leave=False, unit='b'):
            gs, _ = sess.run([g.global_step, g.train_op])

            # Write checkpoint files at every 1k steps
            if gs % 1000 == 0:
                sv.saver.save(sess, logdir + '/model_gs_{}'.format(str(gs //
1000).zfill(3) + "k"))

                if num==1:
                    # plot alignment
                    alignments = sess.run(g.alignments)
                    plot_alignment(alignments[0], str(gs // 1000).zfill(3) +
"k", logdir)

            # break
            if gs > hp.num_iterations: break

print("Done")
```

Step 5 Build predict process.

```
import os

from hyperparams import Hyperparams as hp
import numpy as np
import tensorflow as tf
from train import Graph
from utils import *
from data_load import load_data
from scipy.io.wavfile import write
from tqdm import tqdm

def synthesize():
    # Load data
    L = load_data("synthesize")

    # Load graph
    g = Graph(mode="synthesize"); print("Graph loaded")

    with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())

        # Restore parameters
        var_list = tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES,
'Text2Mel')
        saver1 = tf.train.Saver(var_list=var_list)
        saver1.restore(sess, tf.train.latest_checkpoint(hp.logdir + "-1"))
        print("Text2Mel Restored!")

        var_list = tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES, 'SSRN')
+ \
        tf.get_collection(tf.GraphKeys.GLOBAL_VARIABLES, 'gs')
```

```
saver2 = tf.train.Saver(var_list=var_list)
saver2.restore(sess, tf.train.latest_checkpoint(hp.logdir + "-2"))
print("SSRN Restored!")

# Feed Forward
## mel
Y = np.zeros((len(L), hp.max_T, hp.n_mels), np.float32)
prev_max attentions = np.zeros((len(L),), np.int32)
for j in tqdm(list(range(hp.max_T))):
    _gs, _Y, _max attentions, _alignments = \
        sess.run([g.global_step, g.Y, g.max attentions, g.alignments],
                  {g.L: L,
                   g.mels: Y,
                   g.prev_max attentions: prev_max attentions})
    Y[:, j, :] = _Y[:, j, :]
    prev_max attentions = _max attentions[:, j]

# Get magnitude
Z = sess.run(g.Z, {g.Y: Y})

# Generate wav files
if not os.path.exists(hp.sampledir): os.makedirs(hp.sampledir)
for i, mag in enumerate(Z):
    print("Working on file", i+1)
    wav = spectrogram2wav(mag)
    write(hp.sampledir + "/" + "{}.wav".format(i+1), hp.sr, wav)

if __name__ == '__main__':
    synthesize()
    print("Done")
```

2.5 Summary

This chapter describes how to implement TTS application. In this experiment, the model we use to tackle these tasks is the deep neural network model, namely Seq2Seq with attention. Tensorflow is the framework used in this experiment to build the deep neural network and to process the training of this model. And the every detail of the whole process is displayed in the code.

3 ASR

3.1 Introduction to the Experiment

ASR is the technology used to convert speech into text. The tasks of ASR can be roughly divided into two branches; the first branch includes the isolated words speech recognition tasks which are the simplest task in ASR and can be handled by traditional methods, such as HMM-GMM model; the other branch includes the continuous words speech recognition tasks which are more difficult than isolated words recognition especially when the amount of words is very large, and these tasks are often handled by traditional methods when the amount of words is not that large, but when this condition cannot be satisfied the deep neural network model is often used to tackle these problems, such as RNN, LSTM, Seq2Seq model and so on.

In this experiment, we focus on the second branch of ASR tasks. And the model we use to tackle these tasks is the deep neural network model, namely BiRNN. Of course, there are many other models we can use, we choose this model because it has been proved to have good performance and is easy for new learners to understand.

Tensorflow is the framework used in this experiment to build the deep neural network and to process the training of this model.

3.2 Purpose

This experiment introduces the detailed steps of the whole ASR task, such as how to preprocess the raw data, how to label the raw data and how to build and train the model in tensorflow. Because the amount of code is large and the modules are embedded into each other, especially the training process will take a lot of time, so our main purpose of this experiment is to help new learners to understand the whole framework and process of ASR implemented by tensorflow and after this experiment learners can build their own models to tackle specific ASR applications.

3.3 Knowledge Required

This experiment requires knowledge are as follows:

- Syntactical basis of the Python language and hands-on operation capability
- Understanding the use of tensorflow
- Have basic knowledge of the pipeline of automatic speech recognition
- Understand basic concepts and process of machine learning
- Understand basic knowledge of deep neural network

3.4 Procedure

For the readability and compact style of code, we divide the whole process of this project into two steps. In first step, we build the modules that will be used in many places. In second step, we build the whole train and predict process based on the modules we have built in first step.

3.4.1 Build Basic Modules

Step 1 **Build the data reading function.**

Code:

```
def get_wavs_labels(wav_path, label_file):
    # get the paths of train files

    wav_files = []
    for (dirpath, dirnames, filenames) in os.walk(wav_path):
        for filename in filenames:
            if filename.endswith('.wav') or filename.endswith('.WAV'):
                filename_path = os.sep.join([dirpath, filename])
                if os.stat(filename_path).st_size < 24: # eliminate small files
                    continue
                wav_files.append(filename_path)
    # print(wav_files)

    labels_dict = {}
    with open(label_file, 'r') as f:
        for label in f:
            label = label.strip('\r\n')
            # label = label.replace("")
            label_id = label.split(':', 1)[0]
            # print(label_id)
            label_text = label.split(':', 1)[1]
```

```
#         print(label_text)
        labels_dict[label_id] = label_text
# get the text result of speech file
labels = []
new_wav_files = []
for wav_file in wav_files:
    wav_id = os.path.basename(wav_file).split('.')[0]
#     print(wav_id)
    if wav_id in labels_dict:
        labels.append(labels_dict[wav_id])
        new_wav_files.append(wav_file)
    return new_wav_files, labels
wav_path="D:\\python project\\HCIP-AI\\data"
label_file="D:\\python project\\HCIP-AI\\data\\1.txt"
wav_files, labels = get_wavs_labels(wav_path,label_file)
print(wav_files[0], labels[0])
print("wavs:",len(wav_files),"labels:",len(labels))
```

Result:

```
D:\python project\HCIP-
AI\data\f480b8a93bf84b7f74c141284a71c39ff47d264a75dc905dc918286fb67f033359520
6ff953a27b8049c7ec09ea895aa66d1cd4f7547535167d3d7901d12feab.wav
Unfortunately, nobody can warrant the sanctions that will have an effect on
the community.
wavs: 1 labels: 1
```

Step 2 Build the function transform vector to word

```
#set constant parameters
SPACE_TOKEN = '<space>'
SPACE_INDEX = 0
FIRST_INDEX = ord('a') - 1

# transfer sparse matrix to words
def sparse_tuple_to_texts_ch(tuple,words):
    indices = tuple[0]
    values = tuple[1]
    results = [''] * tuple[2][0]
    for i in range(len(indices)):
        index = indices[i][0]
        c = values[i]

        c = ' ' if c == SPACE_INDEX else words[c]
        results[index] = results[index] + c
    return results

# transfer dense matrix to words
def ndarray_to_text_ch(value,words):
    results = ''
    for i in range(len(value)):
```

```
        results += words[value[i]]
    return results.replace('`', ' ')

# transfer text labels to array
def readalltxt(txt_files):
    labels = []
    for txt_file in txt_files:
        # load the text file and transfer it to array
        target = get_ch_label(txt_file)
        labels.append(target)
    return labels
```

Step 3 Transfer dense matrix to sparse matrix.

```
def sparse_tuple_from(sequences, dtype=np.int32):

    indices = []
    values = []

    for n, seq in enumerate(sequences):
        indices.extend(zip([n] * len(seq), range(len(seq))))
        values.extend(seq)

    indices = np.asarray(indices, dtype=np.int64)
    values = np.asarray(values, dtype=dtype)
    shape = np.asarray([len(sequences), indices.max(0)[1] + 1],
dtype=np.int64)

    return indices, values, shape
```

Step 4 Build the speech feature extraction function.

```
# extract MFCC feature from speech file
def audiofile_to_input_vector(audio_filename, numcep, numcontext):

    # load wavfile
    fs, audio = wav.read(audio_filename)

    # get MFCC coefficient
    orig_inputs = mfcc(audio, samplerate=fs, numcep=numcep)
    orig_inputs = orig_inputs[:, :2]

    train_inputs = np.array([], np.float32)
    train_inputs.resize((orig_inputs.shape[0], numcep + 2 * numcep *
numcontext))

    empty_mfcc = np.array([])
    empty_mfcc.resize((numcep))

    # prepare training data
    time_slices = range(train_inputs.shape[0])
```

```

context_past_min = time_slices[0] + numcontext
context_future_max = time_slices[-1] - numcontext
for time_slice in time_slices:
    # pad 0 for the first 9 positions
    need_empty_past = max(0, (context_past_min - time_slice))
    empty_source_past = list(empty_mfcc for empty_slots in
range(need_empty_past))
    data_source_past = orig_inputs[max(0, time_slice -
numcontext):time_slice]
    assert(len(empty_source_past) + len(data_source_past) == numcontext)

    # pad 0 for the last 9 positions
    need_empty_future = max(0, (time_slice - context_future_max))
    empty_source_future = list(empty_mfcc for empty_slots in
range(need_empty_future))
    data_source_future = orig_inputs[time_slice + 1:time_slice + numcontext
+ 1]
    assert(len(empty_source_future) + len(data_source_future) ==
numcontext)

    if need_empty_past:
        past = np.concatenate((empty_source_past, data_source_past))
    else:
        past = data_source_past

    if need_empty_future:
        future = np.concatenate((data_source_future, empty_source_future))
    else:
        future = data_source_future

    past = np.reshape(past, numcontext * numcep)
    now = orig_inputs[time_slice]
    future = np.reshape(future, numcontext * numcep)

    train_inputs[time_slice] = np.concatenate((past, now, future))
    assert(len(train_inputs[time_slice]) == numcep + 2 * numcep *
numcontext)

    # standardize the data with normal distribution
    train_inputs = (train_inputs - np.mean(train_inputs)) /
np.std(train_inputs)
    return train_inputs

```

Step 5 Build the function convert speech feature to matrix

```

def get_audio_and_transcriptch(txt_files, wav_files, n_input,
                             n_context, word_num_map, txt_labels=None):

    audio = []
    audio_len = []
    transcript = []
    transcript_len = []
    if txt_files!=None:
        txt_labels = txt_files

```

```
for txt_obj, wav_file in zip(txt_labels, wav_files):
    # load speech file and transfer it to MFCC feature
    audio_data = audiofile_to_input_vector(wav_file, n_input, n_context)
    audio_data = audio_data.astype('float32')

    audio.append(audio_data)
    audio_len.append(np.int32(len(audio_data)))

    # load the text lables and transfer it to array
    target = []
    if txt_files!=None: #txt_obj is file
        target = get_ch_lable_v(txt_obj,word_num_map)
    else: #txt_obj is labels
        target = get_ch_lable_v(None,word_num_map,txt_obj)

    transcript.append(target)
    transcript_len.append(len(target))

audio = np.asarray(audio)
audio_len = np.asarray(audio_len)
transcript = np.asarray(transcript)
transcript_len = np.asarray(transcript_len)
return audio, audio_len, transcript, transcript_len
```

Step 6 Build the function transform word to vector

```
def get_ch_lable_v(txt_file,word_num_map,txt_label=None):

    words_size = len(word_num_map)

    to_num = lambda word: word_num_map.get(word, words_size)

    if txt_file!= None:
        txt_label = get_ch_lable(txt_file)

    labels_vector = list(map(to_num, txt_label))
    return labels_vector
```

Step 7 Build pad function for alignment

```
def pad_sequences(sequences, maxlen=None, dtype=np.float32,
                  padding='post', truncating='post', value=0.):

    lengths = np.asarray([len(s) for s in sequences], dtype=np.int64)

    nb_samples = len(sequences)
    if maxlen is None:
        maxlen = np.max(lengths)

    sample_shape = tuple()
    for s in sequences:
        if len(s) > 0:
```

```

        sample_shape = np.asarray(s).shape[1:]
        break

    x = (np.ones((nb_samples, maxlen) + sample_shape) * value).astype(dtype)
    for idx, s in enumerate(sequences):
        if len(s) == 0:
            continue # find empty list
        if truncating == 'pre':
            trunc = s[-maxlen:]
        elif truncating == 'post':
            trunc = s[:maxlen]
        else:
            raise ValueError('Truncating type "%s" not understood' %
truncating)

        # check the shape of trunc
        trunc = np.asarray(trunc, dtype=dtype)
        if trunc.shape[1:] != sample_shape:
            raise ValueError('Shape of sample %s of sequence at position %s \
is different from expected shape %s' %
(trunc.shape[1:], idx, sample_shape))

        if padding == 'post':
            x[idx, :len(trunc)] = trunc
        elif padding == 'pre':
            x[idx, -len(trunc):] = trunc
        else:
            raise ValueError('Padding type "%s" not understood' % padding)
    return x, lengths

```

Step 8 Build BiRNN model

```

# set parameters
b_stddev = 0.046875
h_stddev = 0.046875
n_hidden = 1024
n_hidden_1 = 1024
n_hidden_2 = 1024
n_hidden_5 = 1024
n_cell_dim = 1024
n_hidden_3 = 2 * 1024
relu_clip = 20

# define the architecture
def BiRNN_model( batch_x, seq_length, n_input,
n_context,n_character ,keep_dropout):
    batch_x_shape = tf.shape(batch_x)

    batch_x = tf.transpose(batch_x, [1, 0, 2])

    batch_x = tf.reshape(batch_x,[-1, n_input + 2 * n_input * n_context])

# construct the first 3 fully connected layers
# use clipped RELU activation and dropout.

```

```
# construct the first layer
with tf.name_scope('fc1'):
    b11 = variable_on_cpu('b11', [n_hidden_1],
tf.random_normal_initializer(stddev=b_stddev))
    h1 = variable_on_cpu('h1', [n_input + 2 * n_input * n_context,
n_hidden_1],
                                tf.random_normal_initializer(stddev=h_stddev))
    layer_1 = tf.nn.relu(tf.add(tf.matmul(batch_x, h1), b11)),
relu_clip)
    layer_1 = tf.nn.dropout(layer_1, keep_dropout)

# construct the 2d layer
with tf.name_scope('fc2'):
    b2 = variable_on_cpu('b2', [n_hidden_2],
tf.random_normal_initializer(stddev=b_stddev))
    h2 = variable_on_cpu('h2', [n_hidden_1, n_hidden_2],
tf.random_normal_initializer(stddev=h_stddev))
    layer_2 = tf.nn.relu(tf.add(tf.matmul(layer_1, h2), b2)),
relu_clip)
    layer_2 = tf.nn.dropout(layer_2, keep_dropout)

# construct the 3d layer
with tf.name_scope('fc3'):
    b3 = variable_on_cpu('b3', [n_hidden_3],
tf.random_normal_initializer(stddev=b_stddev))
    h3 = variable_on_cpu('h3', [n_hidden_2, n_hidden_3],
tf.random_normal_initializer(stddev=h_stddev))
    layer_3 = tf.nn.relu(tf.add(tf.matmul(layer_2, h3), b3)),
relu_clip)
    layer_3 = tf.nn.dropout(layer_3, keep_dropout)

# construct bidirectional RNN
with tf.name_scope('lstm'):
    # Forward direction cell:
    lstm_fw_cell = tf.nn.rnn.BasicLSTMCell(n_cell_dim,
forget_bias=1.0, state_is_tuple=True)
    lstm_fw_cell =
tf.nn.rnn.DropoutWrapper(lstm_fw_cell, input_keep_prob=keep_dropout)
    # Backward direction cell:
    lstm_bw_cell = tf.nn.rnn.BasicLSTMCell(n_cell_dim,
forget_bias=1.0, state_is_tuple=True)
    lstm_bw_cell =
tf.nn.rnn.DropoutWrapper(lstm_bw_cell, input_keep_prob=keep_dropout)

    layer_3 = tf.reshape(layer_3, [-1, batch_x_shape[0], n_hidden_3])

    outputs, output_states =
tf.nn.bidirectional_dynamic_rnn(cell_fw=lstm_fw_cell,
                                cell_bw=lstm_bw_cell,
                                inputs=layer_3,
                                dtype=tf.float32,
                                time_major=True,
```

```
sequence_length=seq_length)

# connect bidirectional outputs
outputs = tf.concat(outputs, 2)

outputs = tf.reshape(outputs, [-1, 2 * n_cell_dim])

with tf.name_scope('fc5'):
    b5 = variable_on_cpu('b5', [n_hidden_5],
tf.random_normal_initializer(stddev=b_stddev))
    h5 = variable_on_cpu('h5', [(2 * n_cell_dim), n_hidden_5],
tf.random_normal_initializer(stddev=h_stddev))
    layer_5 = tf.nn.relu(tf.add(tf.matmul(outputs, h5), b5)),
relu_clip)
    layer_5 = tf.nn.dropout(layer_5, keep_dropout)

with tf.name_scope('fc6'):
    # softmax layer for classification
    b6 = variable_on_cpu('b6', [n_character],
tf.random_normal_initializer(stddev=b_stddev))
    h6 = variable_on_cpu('h6', [n_hidden_5, n_character],
tf.random_normal_initializer(stddev=h_stddev))
    layer_6 = tf.add(tf.matmul(layer_5, h6), b6)

# reshape [n_steps*batch_size, n_character] to [n_steps, batch_size,
n_character].
layer_6 = tf.reshape(layer_6, [-1, batch_x_shape[0], n_character])
return layer_6
```

Step 9 Build the function calling on CPU

```
def variable_on_cpu(name, shape, initializer):
    # set the scope of cpu
    with tf.device('/cpu:0'):
        # construct var
        var = tf.get_variable(name=name, shape=shape, initializer=initializer)
    return var
```

3.4.2 Build The Whole Training Process

Step 1 Import Modules

```
import numpy as np
import time
import tensorflow as tf
from tensorflow.python.ops import ctc_ops
from tensorflow.python.tools.inspect_checkpoint import
print_tensors_in_checkpoint_file
from collections import Counter
```

Step 2 Import Functions

Code:

```
sparse_tuple_to_texts_ch = speech_utils.sparse_tuple_to_texts_ch
ndarray_to_text_ch = speech_utils.ndarray_to_text_ch
get_audio_and_transcriptch = speech_utils.get_audio_and_transcriptch
pad_sequences = speech_utils.pad_sequences
sparse_tuple_from = speech_utils.sparse_tuple_from
get_wavs_labels = speech_utils.get_wavs_labels
BiRNN_model = speech_utils.BiRNN_model
variable_on_cpu = speech_utils.variable_on_cpu
dir(speech_utils)
```

Result:

```
['BiRNN_model',
 'FIRST_INDEX',
 'SPACE_INDEX',
 'SPACE_TOKEN',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'audiofile_to_input_vector',
 'b_stddev',
 'get_audio_and_transcriptch',
 'get_ch_label_v',
 'get_wavs_labels',
 'h_stddev',
 'mfcc',
 'n_cell_dim',
 'n_hidden',
 'n_hidden_1',
 'n_hidden_2',
 'n_hidden_3',
 'n_hidden_5',
 'ndarray_to_text_ch',
 'np',
 'os',
 'pad_sequences',
 'readalltxt',
 'relu_clip',
 'sparse_tuple_from',
 'sparse_tuple_to_texts_ch',
 'tf',
 'variable_on_cpu',
 'wav']
```

Step 3 Define Parameters and Paths

```
tf.reset_default_graph()

# define paths parameters
# wav_path is wave file path and label_file is label file path
# wav_path="data_thchs30_10/train"
# label_file="data_thchs30_10/train.word.txt"
wav_path="D:\\python project\\HCIP-AI\\data"
label_file="D:\\python project\\HCIP-AI\\1.txt"
```

Step 4 Import Speech Data

```
wav_files, labels = get_wavs_labels(wav_path, label_file)
print(wav_files[0], labels[0])
print("wavs:", len(wav_files), "labels:", len(labels))
```

Step 5 Batch Data Generting Function

```
def next_batch(labels, start_idx = 0, batch_size=1, wav_files = wav_files):
    filesize = len(labels)
    end_idx = min(filesize, start_idx + batch_size)
    idx_list = range(start_idx, end_idx)
    txt_labels = [labels[i] for i in idx_list]
    wav_files = [wav_files[i] for i in idx_list]
    (source, audio_len, target, transcript_len) =
    get_audio_and_transcriptch(None,
                                wav_files,
                                n_input,

    n_context, word_num_map, txt_labels )

    start_idx += batch_size
    # test the size of start_idx
    if start_idx >= filesize:
        start_idx = -1

    # padding for source files
    source, source_lengths = pad_sequences(source)
    sparse_labels = sparse_tuple_from(target)

    return start_idx, source, source_lengths, sparse_labels
```

Step 6 Define Placeholders

```
input_tensor = tf.placeholder(tf.float32, [None, None, n_input + (2 * n_input
* n_context)], name='input')
#lables
targets = tf.sparse_placeholder(tf.int32, name='targets')

#sequence length
seq_length = tf.placeholder(tf.int32, [None], name='seq_length')
keep_dropout= tf.placeholder(tf.float32)
```

Step 7 Define final operation

```
logits = BiRNN_model( input_tensor, tf.to_int64(seq_length), n_input,
n_context, words_size +1, keep_dropout)
```

Step 8 Define loss function and optimizer

```
#define loss function based on ctc loss
avg_loss = tf.reduce_mean(ctc_ops.ctc_loss(targets, logits, seq_length))

#choose AdamOptimizer as optimizer
learning_rate = 0.0001
optimizer =
tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(avg_loss)
```

Step 9 Define metrics

```
with tf.name_scope("decode"):
    decoded, log_prob = ctc_ops.ctc_beam_search_decoder( logits, seq_length,
merge_repeated=False)

with tf.name_scope("accuracy"):
    distance = tf.edit_distance( tf.cast(decoded[0], tf.int32), targets)
    # label error rate (accuracy)
    ler = tf.reduce_mean(distance, name='label_error_rate')
```

Step 10 Model setting

```
savedir = r"D:\\python project\\TensorFlow Jupyter\\yuyinch_10_10"

epochs = 100 #set the training epoch

# setting for the saver
saver = tf.train.Saver(max_to_keep=1)

# construct a session
sess = tf.Session()

sess.run(tf.global_variables_initializer())

kpt = tf.train.latest_checkpoint(savedir)
print("kpt:", kpt)
startepo= 0
if kpt!=None:
    saver.restore(sess, kpt)
    ind = kpt[::-1].find("-")
    print(ind)
    startepo = int(kpt[-ind:])
    print(startepo)

startepo= 0
```

```
saver.restore(sess, savedir)
ind = kpt.find("-")
startepo = int(kpt[ind+1:])
print(startepo)
```

Step 11 Modle training

```
# set parameters
keep_dropout_rate=0.95
section = '\n{0:=^40}\n'
print(section.format('Run training epoch'))

# training process
train_start = time.time() # record the starting time
for epoch in range(epochs):
    epoch_start = time.time()
    if epoch<startepo:
        continue

    print("epoch start:",epoch+1)
    print("total epochs: ",epochs)

    # print the details of every epoch
    n_batches_per_epoch = int(np.ceil(len(labels) / batch_size))
    print("total loop ",n_batches_per_epoch,"in one epoch ","batch_size","items
in one loop\n")

    train_cost = 0
    train_ler = 0
    next_idx =0

    # batch traing process
    for batch in range(n_batches_per_epoch):# the number of batches
        # get data
        next_idx,source,source_lengths,sparse_labels =
next_batch(labels,next_idx ,batch_size)
        feed = {input_tensor: source, targets: sparse_labels,seq_length:
source_lengths,keep_dropout:keep_dropout_rate}

        # avg_loss optimizer ;
        batch_cost, _ = sess.run([avg_loss, optimizer], feed_dict=feed )
        train_cost += batch_cost

        if (batch +1)%2 == 0:
            print('loop:',batch, 'Train cost: ', train_cost/(batch+1))
            feed2 = {input_tensor: source, targets: sparse_labels,seq_length:
source_lengths,keep_dropout:1.0}

            d,train_ler = sess.run([decoded[0],ler], feed_dict=feed2)
            dense_decoded = tf.sparse_tensor_to_dense( d, default_value=-
1).eval(session=sess)
            dense_labels = sparse_tuple_to_texts_ch(sparse_labels,words)
```

```
counter =0
print('Label err rate: ', train_ler)
for orig, decoded_arr in zip(dense_labels, dense_decoded):
    # convert to strings
    decoded_str = ndarray_to_text_ch(decoded_arr, words)
    print(' file {}'.format( counter))
    print('Original: {}'.format(orig))
    print('Decoded: {}'.format(decoded_str))
    counter=counter+1
    break

# print the results of every epoch and save the training model
epoch_duration = time.time() - epoch_start

log = 'Epoch {}/ {}, train_cost: {:.3f}, train_ler: {:.3f}, time: {:.2f}
sec'
print(log.format(epoch , epochs, train_cost, train_ler, epoch_duration))
saver.save(sess, savedir+"yuyinch.cpkt", global_step=epoch)

#print train_duration
train_duration = time.time() - train_start
print('Training complete, total duration: {:.2f} min
\n'.format(train_duration / 60))

#close session
sess.close()
```

3.5 Summary

This chapter describes how to implement ASR application. In this experiment, we focus on the second branch of ASR tasks that is the continuous words recognition. And the model we use to tackle these tasks is the deep neural network model, namely BiRNN.

Tensorflow is the framework used in this experiment to build the deep neural network and to process the training of this model. And the every detail of the whole process is displayed in the code.

Huawei HCIP-AI Certification Training

HCIP-AI EI Developer

Natural Language Processing

Experiment Guide

ISSUE: 1.0



HUAWEI TECHNOLOGIES CO., LTD.

Copyright © Huawei Technologies Co., Ltd. 2019. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <http://e.huawei.com>

Huawei Certificate System

Huawei Certification follows the "platform + ecosystem" development strategy, which is a new collaborative architecture of ICT infrastructure based on "Cloud-Pipe-Terminal". Huawei has set up a complete certification system consisting of three categories: ICT infrastructure certification, Platform and Service certification and ICT vertical certification, and grants Huawei certification the only all-range technical certification in the industry.

Huawei offers three levels of certification: Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP), and Huawei Certified ICT Expert (HCIE).

The fourth revolution of science and technology, driven by ICT networks and AI, is leading us towards an intelligent world where all things can sense, all things are connected, and all things are intelligent. In October 2018, Huawei Rotating Chairman Eric Xu announced Huawei's AI strategy, as well as its full-stack, all-scenario AI portfolio. The company's full-stack, all-scenario AI portfolio is designed to provide powerful support for Huawei Cloud EI and HiAI. Backed by its AI portfolio, Huawei Cloud EI will be able to deliver a full-stack portfolio for enterprise and government customers, and HiAI will provide a full-stack portfolio for smart devices. HiAI services are deployed on Huawei Cloud EI. However, the gap of AI talent in this world is very large. To seize this historic opportunity and accelerate the construction of the AI talent ecosystem, we hereby release the HCIP-AI EI Developer V1.0 certification to develop talents who have acquired basic theoretical knowledge about image processing, speech processing, and natural language processing (NLP) and who are able to use Huawei enterprise AI solutions (such as Huawei cloud EI) and general open-source frameworks to conduct development and innovation.

HCIP-AI EI Developer V1.0 is a kind of engineer career certification for ICT practitioners, Huawei's users, Huawei engineers, Huawei partners' engineers, and college students. The content of HCIP-AI EI Developer V1.0 includes but is not limited to: deep feedforward networks, image processing theory and applications, speech processing theory and applications, NLP theory and applications, the image processing experiment, the speech processing experiment, and the NLP experiment.

About This Document

Overview

This document is an HCIP-AI certification training course. It is intended for trainees who are preparing for HCIP-AI tests or readers who want to know about AI basics. After understanding this document, you will be able to perform natural language processing (NLP), including word segmentation, text feature extraction, and vectorization. The experiments are performed based on open-source datasets by using the Python language. Not only specific implementation processes but also effects of the NLP functions are offered.

Description

This document contains the following three experiments and presents their effects to help trainees or readers build the AI NLP development capability.

- Experiment 1: To learn the feature extraction of the term frequency-inverse document frequency (TF-IDF) of Python-based text
- Experiment 2: To learn Python-based text Word2Vec processing
- Experiment 3: To learn Python-based text Doc2Vec processing

Background Knowledge Required

To better master the contents of this course, the audience should:

- Have basic knowledge in Python language programming;
- Have basic theoretical knowledge in NLP;
- Have a basic understanding of framework functions.

Experiment Environment Overview

- This experiment environment is developed and compiled based on Python 3.6.
- Anaconda3 (64-bit)

- Jupyter Notebook
- URL for downloading experimental datasets: <https://data-certification.obs.cn-east-2.myhuaweicloud.com/ENG/HCIP-AI%20EI%20Developer/V1.0/NLP.zip>

Contents

Overview	2
Description	2
Background Knowledge Required	2
Experiment Environment Overview	2
1 TF-IDF.....	5
1.1 Experiment Overview	5
1.2 Experiment Objectives	5
1.3 Background Knowledge Required	5
1.4 Module Installation	6
1.5 Experiment Procedure.....	7
1.6 Summary	10
2 Word Embedding: Word2Vec.....	11
2.1 Experiment Overview	11
2.2 Experiment Objectives	11
2.3 Background Knowledge Required	11
2.4 Module Installation	12
2.5 Experiment Procedure.....	13
2.6 Code Implementation	13
2.7 Summary	16
3 Sentence Embedding: Doc2Vec.....	17
3.1 Experiment Overview	17
3.2 Experiment Objectives	17
3.3 Background Knowledge Required	17
3.4 Experiment Procedure.....	18
3.5 Summary	22

1 TF-IDF

1.1 Experiment Overview

The TF-IDF is a statistical method used for measuring the importance of a term in a file of a fileset or corpus. The importance of the term is in direct proportion to the number of times the term appears in the file, and in inverse proportion to its appearance frequency in the corpus. Term frequency-inverse document frequency (TF-IDF) is a common weighted technology used for information retrieval and data mining.

Idea behind TF-IDF: If a term frequently appears in an article, and hardly appears in other articles, the term boasts a high category identification capability, and it is applicable for classification.

That is, if a keyword appears in an article more frequently than it does in other articles, this keyword can better represent this article.

This experiment, in combination with theory, measures the importance of a term in a simple dataset **huawei.txt** in Python using the TF-IDF method. This method can be used in subsequent deep NLP.

1.2 Experiment Objectives

This experiment requires trainees to combine theory with practice and implement TF-IDF in the Python and Scikit-learn frameworks, to deeply understand NLP.

1.3 Background Knowledge Required

To operate this experiment, trainees should:

Have basic theoretical knowledge in the TF-IDF technology of NLP, which is used for extracting features in NLP classification tasks, yielding a satisfactory result.

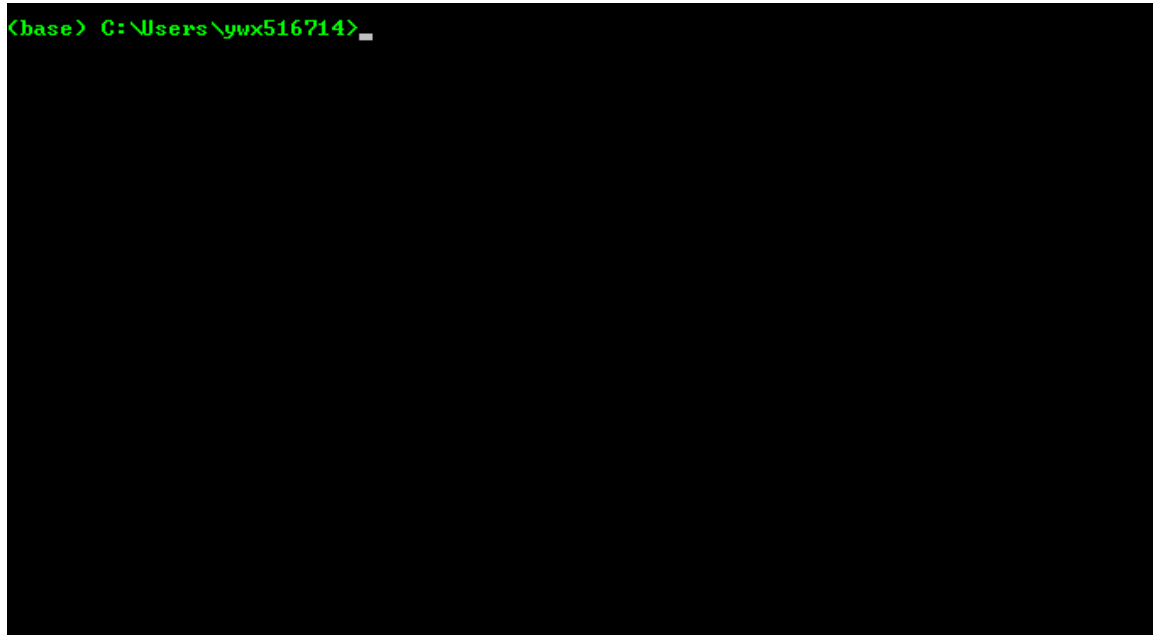
Have experience in the Python language. Understand basic data formats, data types, and related character processing functions of Python, and know about file read/write operations.

Understand the TF-IDF module in the machine learning framework Scikit-learn.

1.4 Module Installation

Install the most popular machine learning framework of Python.

Choose **Start > Anaconda Prompt** to enter the Anaconda system.



```
<base> C:\Users\ywx516714> _
```

Figure 1-1 Enter the Anaconda system

Install the module in either of the methods: Anaconda-based conda install Scikit-learn or Python-based pip install Scikit-learn. Use the former one as an example here.

```
(base) C:\Users\ywx516714>conda install scikit-learn
Solving environment: done

## Package Plan ##

  environment location: D:\ProgramData\Anaconda3

  added / updated specs:
    - scikit-learn

The following packages will be downloaded:

  package                                     |      build
  -----|-----
  scikit-learn-0.20.0                       |  py37heebcf9a_1          5.2 MB  https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/

The following NEW packages will be INSTALLED:

  scikit-learn: 0.20.0-py37heebcf9a_1 https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main

Proceed [y]/n)? y

Downloading and Extracting Packages
scikit-learn-0.20.0 : 5.2 MB : #####
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(base) C:\Users\ywx516714>_
```

Figure 1-2 Install the Scikit-learn module

Install other required frameworks in the similar procedure.

1.5 Experiment Procedure

In the machine learning framework Scikit-learn of Python, the TF-IDF method has been implemented. This experiment is to test and verify the effect in the method in the framework. The experiment procedure is as follows:

Obtain a corpus.

Transform terms into a frequency matrix.

Calculate the TF-IDF weight values.

View the result.

1.5.1 Code Implementation

Step 1 Import the module.

Code:

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
```

Step 2 Obtain data after word segmentation.

The data has been processed with word segmentation based on a self-developed simple dataset. Raw data needs to be segmented before it can be used in this experiment.

Code:

```
#Switch to File Corresponding Path
with open(r"C:\Users\swx640908.CHINA\Desktop\NLP\data\huawei.txt",
        "r",encoding="utf-8") as f:
    dataset = list(f.readlines())
    print(dataset)
```

Result:

```
[HUAWEI Mate X receives the first 5G CE certificate awarded by TÜV Rheinland,
a global leader in independent inspection services, which indicates the
smartphone has smoothly passed the rigorous test and evaluation and become
the world's first 5G smartphone approved by the Notify Body of the European
Union ... ...(part of the content omitted)']
```

Step 3 Read the stop words.

Read the stop words commonly used in Chinese and English to remove them in the corpus.

Code:

```
with
open(r"C:\Users\swx640908.CHINA\Desktop\NLP\data\stopwords.txt","r",encoding=
"utf-8") as f:
    stopwords = list(f.readlines())
    print(stopwords)
# View the stop words.
print(stopwords)
```

Result:

```
['\u0000--\n', '?\n', '\n', '\n', '»\n', '--\n', 'able\n', 'about\n',
'above\n', 'according\n', 'accordingly\n', 'across\n', 'actually\n',
'after\n', 'afterwards\n', 'again\n', 'against\n', "ain't\n", 'all\n',
'allow\n', 'allows\n', 'almost\n', 'alone\n', 'along\n', 'already\n',
'also\n', 'although\n', 'always\n', 'am\n', 'among\n', 'amongst\n', 'an\n',
'and\n', 'another\n', 'any\n', 'anybody\n', 'anyhow\n', 'anyone\n',
'anything\n', 'anyway\n', 'anyways\n', 'anywhere\n', 'apart\n', 'appear\n',
'appreciate\n', 'appropriate\n', 'are\n', "aren't\n", 'around\n', 'as\n',
'a's\n', 'aside\n', 'ask\n', 'asking\n', 'associated\n', 'at\n',
'available\n', 'away\n', 'awfully\n' (part of the content omitted)']
```

Step 4 Transform the terms into a frequency matrix, and calculate the TF-IDF weight values.

Transform the terms in the text into a frequency matrix, and calculate the TF-IDF weight values of all the terms.

Code:

```
vectorizer=CountVectorizer(stop_words = stopwords, min_df=0 )
# This code comment transforms the terms in the text into a frequency matrix.
The matrix element a[i][j] indicates the frequency of the term j in the text
category i.

transformer=TfidfTransformer()
# This code comment measures the TF-IDF weight value of each term.
# tfidf=transformer.fit_transform(vectorizer.fit_transform(dataset))
# The first fit_transform calculates the TF-IDF weight values, and the second
fit_transform
#transforms the terms into a frequency matrix.

tfidf=transformer.fit_transform(vectorizer.fit_transform(dataset))
# The first fit_transform calculates the TF-IDF weight values, and the second
fit_transform #transforms the terms into a frequency matrix.

word=vectorizer.get_feature_names()
# Obtain all terms in the Bag of Words model.
print("word:",word)
```

Result:

```
word: ['5g', 'awarded', 'by', 'ce', 'certificate', 'first', 'huawei', 'mate',
'receives', 'rheinland', 'the', 'täv']
```

Step 5 View all keywords of the text and their positions.

Code:

```
print(vectorizer.vocabulary_)
```

Result:

```
{'huawei': 6, 'mate': 7, 'receives': 8, 'the': 10, 'first': 5, '5g': 0, 'ce':
3, 'certificate': 4, 'awarded': 1, 'by': 2, 'täv': 11, 'rheinland': 9}
```

Step 6 Extract the TF-IDF matrix.

The element $a[i][j]$ indicates the TF-IDF weight value of the term j in the text category i .

Code:

```
weight=tfidf.toarray()
print("weight:",weight)
```

Result:

```
weight: [[0.28867513 0.28867513 0.28867513 0.28867513 0.28867513 0.28867513
0.28867513 0.28867513 0.28867513 0.28867513 0.28867513 0.28867513]]
```

Step 7 Print out the TF-IDF weight values of terms for each text category.

Print out the TF-IDF weight values of terms for each text category. The first **for** traverses all text categories and the second **for** traverses weights of terms in a certain text category.

Code:

```
for i in range(len(weight)):
    print(u" The TF-IDF weight values of the terms for category ",i,u" text
are provided:")
    for j in range(len(word)):
        print("    ",word[j],weight[i][j])
```

Result:

```
The TF-IDF weight values of the terms for in text category 0 are provided:
5g 0.2886751345948129
awarded 0.2886751345948129
by 0.2886751345948129
ce 0.2886751345948129
certificate 0.2886751345948129
first 0.2886751345948129
huawei 0.2886751345948129
mate 0.2886751345948129
receives 0.2886751345948129
rheinland 0.2886751345948129
the 0.2886751345948129
t  v 0.2886751345948129
```

----End

1.6 Summary

This experiment implements the TF-IDF method in the Scikit-learn framework, and calculates the TF-IDF weight value of each term in the text. A great weight value means a great impact of the term on the text, and a small weight value means a small impact.

2 Word Embedding: Word2Vec

2.1 Experiment Overview

When the open-source word vector calculation tool Word2Vec was released by Google in 2013, it arose the attention of the industry and academia. Word2Vec enables efficient training on millions of dictionaries and hundreds of millions of datasets. The word vector (word embedding) obtained after training by this tool can precisely measure similarities between words. Word2Vec is not a deep learning algorithm. It has only a shallow neural network, which contains two models: CBOW and Skip-gram.

This experiment implements Word2Vec on a self-developed simple dataset in the Python and Gensim frameworks, and measures similarities between related words.

2.2 Experiment Objectives

This experiment requires trainees to implement the CBOW and Skip-gram models of Word2Vec on the dataset in the Python and Gensim frameworks. Because of the representation of Word2Vec common words of text vectorization, Word2Vec can be used in subsequent deep NLP tasks.

2.3 Background Knowledge Required

To operate this experiment, trainees should:

Have basic knowledge in the Word2Vec technology. Word2Vec is used for word vectorization, yielding a satisfactory result in subsequent tasks.

Have experience in the Python language. Understand basic data formats, data types, and related character processing functions of Python, and know about file read/write operations.

Be familiar with the Word2Vec submodule in the Gensim framework.

2.4 Module Installation

Install the Gensim framework.

Gensim is an open-source third-party Python toolkit which is used for unsupervised learning of theme vector expressions of the hidden layer from the raw unstructured text. Gensim supports multiple types of theme model algorithms (including TF-IDF, LSA, LDA and word2vec) and streaming training, and provides APIs for executing common tasks such as similarity calculation and information retrieval.

Click **Anaconda Prompt** to enter the Anaconda system.

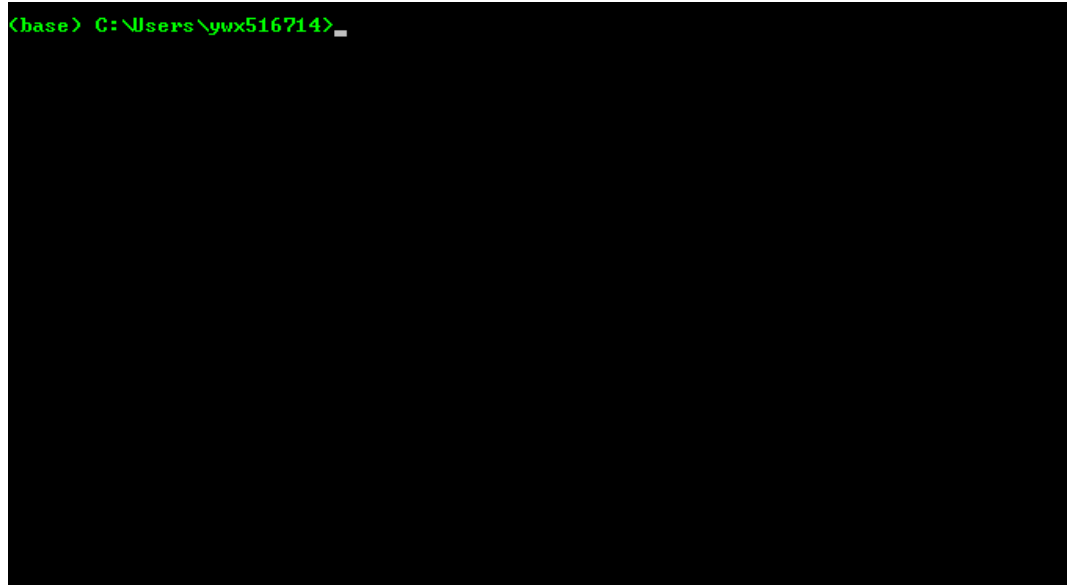


Figure 2-1 enter the Anaconda system

Install the module in either of the methods: Anaconda-based conda install gensim or Python-based pip install gensim. Use the former one as an example here.

```
(base) C:\Users\ywx516714>conda install gensim
Solving environment: done

## Package Plan ##

  environment location: D:\ProgramData\Anaconda3

  added / updated specs:
    - gensim

The following NEW packages will be INSTALLED:

  gensim: 3.4.0-py37hfa6e2cd_0 https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main

Proceed <[y]/n>? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(base) C:\Users\ywx516714>
```

Figure 2-2 Install the gensim module

Install other required Python frameworks in the similar procedure.

2.5 Experiment Procedure

This experiment implements the two models of Word2Vec on a self-developed simple dataset in the Gensim framework, and measures similarities between words. The experiment procedure is as follows:

Prepare data.

Train the models.

Call the models.

Verify the effects.

2.6 Code Implementation

Step 1 Import the module.

Code:

```
import jieba
import os
import gensim
from gensim.models import word2vec, Word2Vec
import logging
# Process the alarm.
import warnings
```

```
warnings.simplefilter('ignore')
```

Step 2 Set paths to store data.

Define the paths to store the corpus and models.

Code:

```
#the purpose of this function is to segment the initial corpus and use it as
a training model.
def preprocessing(old_file):
    print(".....Cutting.....")
    global cut_file    # the file name is saved after word segmentation
    cut_file = old_file + '_cut.txt'

    with open(old_file, 'r') as fi:
        text = fi.read() # obtain text content
        new_text = jieba.cut(text, cut_all=False) # exact mode
        str_out = ' '.join(new_text).replace(', ', ' ').replace('。 ',
        '').replace('? ', ' ').replace('! ', ' ') \
            .replace('“ ', ' ').replace('” ', ' ').replace(':', ' ').replace('...',
        '').replace(' ( ', ' ').replace(' ) ', ' ') \
            .replace('—', ' ').replace('《 ', ' ').replace('》 ', ' ').replace('\ ',
        '').replace('" ', ' ').replace('/', ' ') \
            .replace('\' ', ' ').replace('-', ' ').replace('\n', ' ').replace(')',
        '').replace('(', ' ')
    # Delete punctuation
    print(type(str_out))

    # stopwords
    str_outss = []
    stopwords = [line.strip() for line in
open(r'C:\Users\swx640908.CHINA\Desktop\NLP\data\stopwords.txt', 'r',
encoding='utf-8').readlines() ]
    print(type(stopwords))

    for line in str_out:
        line = line.split()
        str_outss.append([f for f in line if f not in stopwords])
    # write file
    with open(cut_file, 'w', encoding='utf-8') as fo:
        fo.write(str_out)
```

Step 3 Train the models.

Code:

```
# model_file_name is the path of the training corpus.
# save_model is the name of the model saved.
def model_train(train_file_name, save_model_file):
    print("Training.....")
    # Load the corpus.
    sentences = word2vec.Text8Corpus(train_file_name)
    # Train the models, and generate word vector.
```

```
model = Word2Vec(sg=1, size=200, alpha=0.00001, iter=10000, min_count=2)
# Train the Skip-gram model. By default, window is 5.
model.build_vocab(sentences)
model.save(save_model_file)
```

Step 4 Call the models.

Code:

```
#word2vec model training and calling
# Word segmentation
# Determine the path based on the file storing location.
preprocessing(r'C:\Users\swx640908.CHINA\Desktop\NLP\data\huawei.txt') #
Files must first be stored in UTF-8 encoding format.

# Training
# Determine the path based on the file storing location.
save_model_name = r'C:\Users\swx640908.CHINA\Desktop\NLP\data\huawei.model'

model_train(cut_file, save_model_name)
#Determine whether the models exist. If not, retrain the models. If yes,
output is directly given.
if not os.path.exists(save_model_name):    # Determine whether the files
exist.
    model_train(cut_file, save_model_name)
else:
    print('This training model already exists and does not need to be trained
again.')
```

```
# Load the trained models.
model_1 = Word2Vec.load(save_model_name)
# View the models.
print(model_1)
```

Result:

```
This training model already exists and does not need to be trained again.
Word2Vec(vocab=244, size=200, alpha=0.025)
```

Step 5 Measure the similarity of two words.

Enter two words existing in a raw document, and measure the similarity of the two words.

Code:

```
y1 = model_1.similarity("huawei", "safety")
print(u"The similarity is: \n",' ', y1)
```

Result:

```
The similarity is:
```

0.023289131

Step 6 Calculate a list of related words.

Calculate words similar to a certain word in a document, and extract the top 10 words.

Code:

```
# Calculate a list of related words.
y2 = model_1.most_similar("huawei", topn=10) # 10 most relevant
# Machine similarity of similar words
print(u"The most relevant words are:")
for item in y2:
    print("    ",item[0],":", item[1])
```

Result:

```
The most relevant words are:
Huawei : 0.20759014785289764
future : 0.16281606256961823
standards : 0.12602289021015167
that : 0.10039260983467102
scale : 0.10021613538265228
receives : 0.09118469804525375
support : 0.08811673521995544
been : 0.08666529506444931
more : 0.08375560492277145
```

----End

2.7 Summary

This experiment implements the similarity calculation of the Word2Vec method on a self-developed simple dataset in the Gensim framework. Only the Skip-gram Word2Vec model is described here. You can operate the CBOW model by yourself.

3

Sentence Embedding: Doc2Vec

3.1 Experiment Overview

Similar to Word2Vec, Doc2Vec is a sentence vectorization method and includes two models: Distributed Memory (DM) and Distributed Bag of Words (DBOW).

Doc2Vec is also referred to as Paragraph2Vec or sentence embedding. As an unsupervised algorithm, it helps obtain vector expressions of sentences, paragraphs and documents, and expands the functions of Word2Vec. **Doc2Vec can use sentences with different lengths for training.**

This experiment implements Doc2Vec on a self-developed simple dataset in the Gensim framework, and measures similarities between sentences.

3.2 Experiment Objectives

This experiment requires trainees to combine theory with practice and implement Doc2Vec in the Python and Gensim frameworks, to deeply understand NLP.

3.3 Background Knowledge Required

To operate this experiment, trainees should:

Have basic knowledge in the Doc2Vec technology. Doc2Vec is used for sentence vectorization, yielding a satisfactory result in subsequent tasks.

Have experience in the Python language. Understand basic data formats, data types, and related character processing functions of Python, and know about file read/write operations.

Be familiar with the Doc2Vec submodule in the Gensim framework.

3.4 Experiment Procedure

This experiment implements the two models of Doc2Vec on a self-developed dataset in the Gensim framework, and measures similarities between sentences. The experiment procedure is as follows:

Prepare data.

Calculate the Doc2Vec value.

Define the model training function.

Define the model test function.

Define the main function.

3.4.1 Code Implementation

Step 1 **Import the module.**

Enter two sentences existing in a raw document, and measure the similarity of the two sentences.

Code:

```
import jieba
import gensim
import sklearn
import numpy as np
from gensim.models.doc2vec import Doc2Vec, LabeledSentence
# Process the alarm.
import warnings
warnings.simplefilter('ignore')
warnings.simplefilter('always')
warnings.simplefilter('error')
warnings.filterwarnings("ignore")
```

Step 2 **Tag input data.**

The input of the Doc2Vec model should be a TaggedDocument list. Before starting of input data, tag the data.

Code:

```
# text segmentation
old_file = r"C:\Users\swx640908.CHINA\Desktop\NLP\data\txts.txt"
new_file = r"data\txts_cut.txt"

with open(old_file, "r") as f:
    text = f.read()
    print(text)
```

```
with open(new_file, "w", encoding="utf-8") as f:
    for i in text:
        f.write(i)
TaggedgedDocument = gensim.models.doc2vec.TaggedDocument
```

Step 3 Define the data obtainment function.

Define the data obtainment function, run the function, and view the result.

Code:

```
def get_datasest():
    with open("data/texts_cut.txt", "r", encoding="utf-8") as cf:
        docs = cf.readlines()
        print(len(docs))
    x_train = []
    for i, text in enumerate(docs):
        word_list = text.split(' ')
        l = len(word_list)
        word_list[l-1] = word_list[l-1].strip()
        document = TaggedgedDocument(word_list, tags=[i])
        x_train.append(document)
    return x_train
# View the result.
get_datasest()
```

Step 4 Define Doc2Vec.

Code:

```
def getVecs(model, corpus, size):
    vecs = [np.array(model.docvecs[z.tags[0]].reshape(1, size)) for z in corpus]
    return np.concatenate(vecs)
```

Step 5 Define the model training function.

Code:

```
def train(x_train, size=200, epoch_num=100):
    print("training.....")
    model_dm = Doc2Vec(x_train, dm=1, min_count=1, window = 5, vector_size = size, sample=1e-3, negative=5, workers=4)
    model_dm.train(x_train, total_examples=model_dm.corpus_count, epochs=50)
    model_dm.save(r'C:\Users\swx640908.CHINA\Desktop\NLP\model\model_dm')
    return model_dm
```

Step 6 Define the model test function.

Code:

```
def test():
```

```
print("testing.....")
model_dm =
Doc2Vec.load(r"C:\Users\swx640908.CHINA\Desktop\NLP\model\model_dm")
test_text = ['High', 'hopes', 'for', 'snow', 'plowing']
inferred_vector_dm = model_dm.infer_vector(test_text)
print(inferred_vector_dm)
sims = model_dm.docvecs.most_similar([inferred_vector_dm], topn=20)
return sims
```

Code:

```
if __name__ == '__main__':
    x_train = get_data()

    save_model_name = r'C:\Users\swx640908.CHINA\Desktop\NLP\model\model_dm'
    if not os.path.exists(save_model_name): # Determine whether the files
    exist.
        model_dm = train(x_train)
    else:
        print('The training model already exists, and does not need to be
        trained again')

    model_dm = train(x_train)
    sims = test()
    print("sims:",sims)
    for count, sim in sims:
        sentence = x_train[count]
        words = ''
        for word in sentence[0]:
            words = words + word + ' '
        print(words, '-----',sim , '-----
        ',len(sentence[0]))
```

Result:

```
44029
training.....
training.....
testing.....
[ 1.21792126e-02  2.37304214e-02  1.43711641e-02  2.17590965e-02
 -2.42177546e-02 -6.22656010e-02  4.76788580e-02 -3.76284756e-02
 -1.28476471e-02  2.00940035e-02 -8.14704038e-03 -5.25486097e-02
 -3.65001746e-02  4.25280668e-02 -9.91961453e-03 -3.53608057e-02
 -3.73766497e-02  3.52143608e-02  1.93512309e-02 -6.46911236e-03
  4.14997377e-02  1.14495037e-02  2.66781487e-02  1.96467508e-02
 -1.08179385e-02 -4.01965529e-02 -1.30963484e-02  2.02095434e-02
 -6.85250107e-03  5.67306876e-02  8.57827440e-02 -3.55982967e-02
  2.99422182e-02  3.39372382e-02 -7.47787254e-03  2.76217191e-03
 -6.92119002e-02  3.92686687e-02  6.11967221e-02 -4.75075617e-02
 -3.51474583e-02 -2.70418748e-02 -4.50898223e-02 -2.52542198e-02
  6.54226774e-03 -5.14682382e-02 -2.14522779e-02  9.37265530e-03
  7.24302456e-02  1.31152542e-02  6.15672357e-02 -4.69556078e-03
  5.09943627e-02 -1.87107734e-02 -6.94686472e-02 -2.51460336e-02]
```

```
1.44648582e-01 3.84750366e-02 6.57053143e-02 8.14227313e-02
5.58057427e-02 -2.25841794e-02 5.61183728e-02 -2.29374059e-02
-6.90045878e-02 -2.01742239e-02 -4.58984114e-02 -8.50859843e-03
1.16066104e-02 -1.87964533e-02 -2.17472184e-02 7.55883753e-02
4.75960970e-02 2.90594772e-02 -2.07092450e-03 -5.84694631e-02
-2.18964163e-02 2.63798628e-02 -1.70555320e-02 -4.16438794e-03
-4.48973067e-02 -3.65718901e-02 5.04464703e-03 8.45219791e-02
-4.75860573e-03 7.07465559e-02 -4.99042198e-02 -1.97621565e-02
2.87892781e-02 -1.63564794e-02 -5.66919670e-02 -3.30885723e-02
1.17114382e-02 -4.48916666e-02 1.93595566e-04 1.27285058e-02
1.69202231e-03 -3.64423171e-02 -1.49818296e-02 -7.97758400e-02
4.38745483e-05 -5.36401421e-02 -1.47087043e-02 -6.36975095e-02
3.41550186e-02 9.74947959e-03 -1.44520029e-02 -4.61909367e-04
-7.21808746e-02 -3.88580300e-02 8.14109966e-02 -3.94560099e-02
8.79410654e-02 9.63926222e-03 4.59908992e-02 4.55458248e-05
2.28591133e-02 4.44316156e-02 1.67739503e-02 1.53246671e-02
-4.97714803e-02 -5.28839305e-02 -3.22268419e-02 -1.29331890e-02
3.00769191e-02 4.14949730e-02 -3.22365165e-02 -7.52166510e-02
1.98012609e-02 -2.65014754e-03 -2.93836817e-02 1.09305838e-02
2.27258410e-02 3.87894991e-03 5.55930398e-02 1.34629132e-02
-4.61840816e-02 6.04557851e-03 4.30043507e-03 2.23118085e-02
-3.27252760e-03 3.31780082e-03 5.78437652e-03 4.73870374e-02
-3.41983363e-02 -3.06231491e-02 2.12967005e-02 6.44773990e-02
7.35322246e-03 -1.56230070e-02 -8.65432993e-02 -1.97316576e-02
2.75866187e-04 4.87283207e-02 -1.49232242e-02 7.34391510e-02
8.14326629e-02 1.25663392e-02 1.70575436e-02 -2.56433524e-02
5.78162260e-02 -1.65296230e-03 -2.45984015e-03 1.69629343e-02
-2.29643937e-03 -3.20289508e-02 2.70051528e-02 -3.32914442e-02
2.70285401e-02 -3.44775096e-02 -4.88386117e-02 1.50679648e-02
-5.26112784e-03 -2.51380890e-03 -9.86618828e-03 1.05781756e-01
9.46929306e-02 -1.13573857e-01 7.27733150e-02 -1.47498641e-02
8.09536502e-02 3.41392495e-02 1.12058856e-01 -1.95843391e-02
-2.90993378e-02 3.19389701e-02 7.03233480e-02 1.71290208e-02
9.61381849e-03 2.29295213e-02 6.66418970e-02 3.29559594e-02
2.41388511e-02 3.07260384e-03 8.10747743e-02 -2.20663939e-02
6.39309809e-02 -2.23701801e-02 -4.20578383e-02 -7.95688853e-03]
sims: [(27086, 0.438140869140625), (14300, 0.39020562171936035), (17185,
0.38183653354644775), (14249, 0.3772312104701996), (27079,
0.3751215636730194), (35361, 0.37119966745376587), (9958,
0.3668372333049774), (4369, 0.3554239273071289), (9489, 0.35100144147872925),
(22538, 0.3484729528427124), (11952, 0.3463948965072632), (32096,
0.33931005001068115), (41812, 0.3314708471298218), (16298,
0.3285031318664551), (42849, 0.32748422026634216), (32533,
0.32419317960739136), (30720, 0.3237154483795166), (38820,
0.3226546049118042), (30864, 0.3211733102798462), (28980,
0.31860995292663574)]
0.438140869140625 10 ---- "Ice climbing: a winter adventure when the snow
won't fall"
0.39020562171936035 14 ---- "Cadillac's 2011 CTS-V Coupe could become the
hottest American super car on our roads"
0.38183653354644775 10 ---- 'Beware of snow dumps on the 15th street bike
lane'
0.3772312104701996 10 ---- 'Brave the snow to network at the Vertigo Sky
Lounge!'
```

```
0.3751215636730194 11 ---- 'How you may be leaving thousands of dollars on
the table'
0.37119966745376587 10 ---- 'How will the earthquake in Haiti affect Miami
real estate?'
0.3668372333049774 12 ---- 'Rusty Dog sinks in snow but keeps his eye on the
prize'
0.3554239273071289 11 ---- 'Southeast US winter storm possibly in the cards
late this week'
0.35100144147872925 12 ---- 'Natural Balance Pet Foods snow boarding bulldogs
wow crowd at Rose Parade'
0.3484729528427124 11 ---- 'Sunday Forecast: High pressure to dominate the
weather pattern this week'
0.3463948965072632 6 ---- 'Help paying your winter heating bill'
0.33931005001068115 4 ---- 'Contribution: Baltimore versus Haiti'
0.3314708471298218 8 ---- 'Lose the baby fat without leaving the baby'
0.3285031318664551 7 ---- 'Take action put Congress on our Healthcare!'
0.32748422026634216 14 ---- 'Washington: Child care assistance on the
chopping block; could force families deeper into poverty'
0.32419317960739136 8 ---- 'Group calls attention to dog training during
January'
0.3237154483795166 8 ---- 'Report: African children survive better without UN
assistance'
0.3226546049118042 13 ---- 'Carolina Weather 101: How does sea fog form along
the North Carolina coast?'
0.3211733102798462 8 ---- 'Should the American people accept wall street
apologies?'
0.31860995292663574 15 ---- "A tough road ahead for Georgia lawmakers.
Governor Purdue urges 'make the hard choices now.'"
```

----End

3.5 Summary

This experiment implements the similarity calculation of the Doc2Vec method on a self-developed simple dataset in the Gensim framework. Only the DBOW Doc2Vec model is described here. You can operate the DM model by yourself.