【Codelab】基于 AI 通用文字识别的图像搜索，这波操作亮了

我们已经习惯了用手机拍照，记录生活的美好瞬间，但有时候面对浩如烟海的照片，想要快速找到目标图片却显得力不从心。那么，我们能不能开发一款图像搜索工具，通过识别图片上面的文字，匹配用户输入的关键词，帮助用户轻松精准、快速搜索到目标图像呢？

本期我们就为大家带来 HarmonyOS 基于 AI 通用文字识别和分词能力的图像搜索实操项目。该篇 Codelab 不仅汇聚 AI 通用文字识别、AI 分词等技术，还将讲解图片适配及列表展示、用户输入关键词、日志打印等操作，是不是很值得期待！

我们先来了解该应用的功能，用户只需在搜索框中输入关键词，系统即可在图片库中快速匹配含有关键词的图片，并以列表形式呈现，用户还能左右滑动阅览图片。示例效果如下：

接下来，让我们梳理一遍开发要点：

1）编写布局文件

2）添加并展示图片

3）识别图片中的文字

4）提取用户输入的关键词

5）根据关键词匹配图片

6）日志打印

来源：HarmonyOS 开发者微信号 https://mp.weixin.qq.com/s/Oi06hTRUFhlK2PdftE1S7g

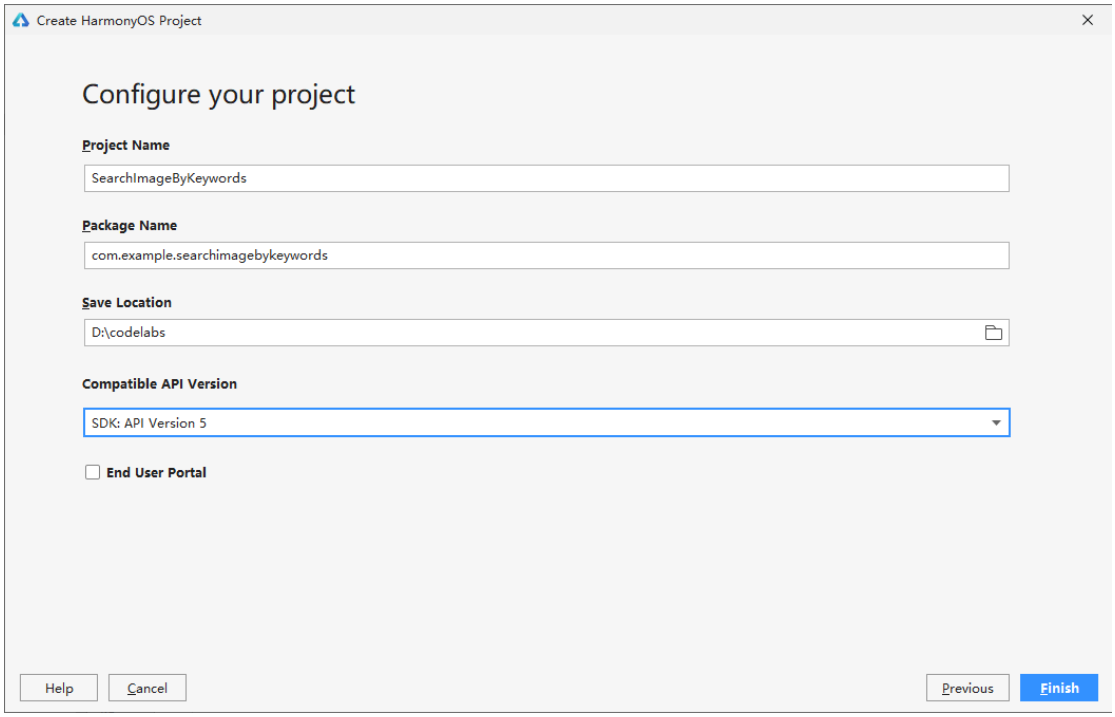
在正式开始敲代码之前，开发者们需要先下载安装 Huawei DevEco Studio，如果对于这个流程不甚熟悉，可以参照官网的教程操作。

Huawei DevEco Studio 安装指南：

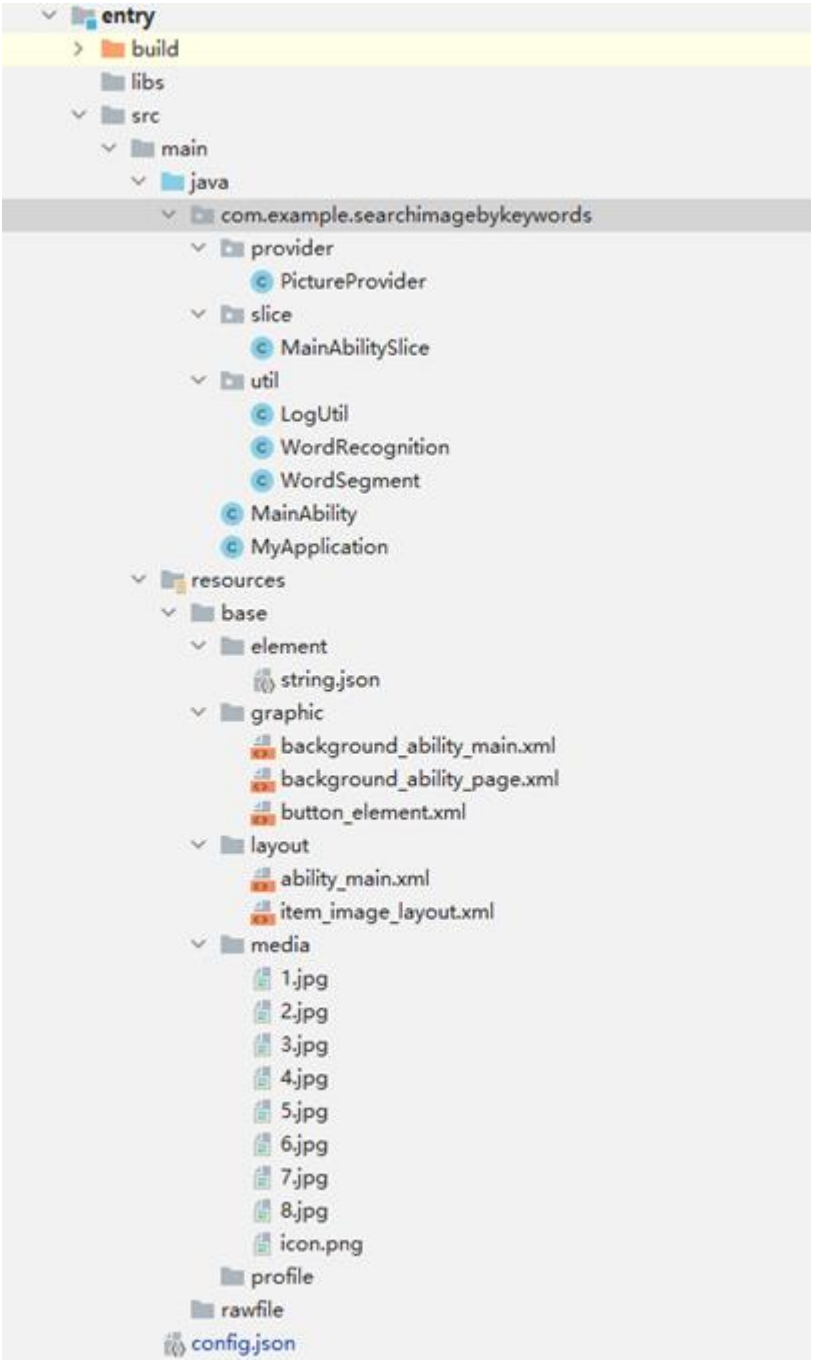
https://developer.harmonyos.com/cn/docs/documentation/doc-guides/software_install-0000001053582415

请注意：本次 Codelab 针对的是步骤拆解和重点讲解，限于篇幅原因不会展示完整代码，

首先,我们在 HUAWEI DevEco Studio 中创建一个 Phone 的 Empty Feature Ability(Java) 模板工程，我们将在这个模板中完成操作。



然后，我们熟悉一下工程的代码结构，如下图：



●provider：PictureProvider 图片适配类，获取所有图片，并将图片放到图片列表中。

●slice：MainAbilitySlice 本示例教程主页面。

●util：工具类

–LogUtil 是日志打印类，对 HiLog 日志进行了封装。

–WordRecognition 是通用文字识别类，对图片中的文字进行识别并保存。

来源：HarmonyOS 开发者微信号 https://mp.weixin.qq.com/s/Oi06hTRUFhlK2PdftE1S7g

-WordSegment 是分词类，对输入文本进行分词。

●MainAbility：主程序入口，DevEco Studio 生成，未添加逻辑，不需变更。

●MyApplication：DevEco Studio 生成，不需变更。

●resources：存放工程使用到的资源文件

-resources\base\element 中存放 DevEco studio 自动生成的配置文件 string.json，不用变更。

-resources\base\graphic 中存放页面样式文件：

●background_ability_page.xml 用于设置界面背景颜色。

●background_ability_main.xml 用于设置界面布局样式。

●button_element.xml 用于设置按钮样式。

-resources\base\layout 中布局文件：

●ability_main.xml 用于展示图片和输入文本。

●item_image_layout.xml 用于设置图片滑动区域图片。

-resources\base\media 下存放图片资源（本教程使用了 8 张.jpg 图片，开发者自行准备；icon.png 由 DevEco Studio 生成不需变更）。

●config.json：配置文件。

编写布局文件

Step 1 ：我们在“resources>base>layout”目录下的 ability_main.xml 文件中，绘制页面布局。

在高度方面,我们采取的是由组件内容（match_content）决定当前组件大小，宽度方面采取的是由父布局(match_parent)决定当前组件大小,整体采取居中布局(vertical_center)，代码如下:

```
<DirectionalLayout
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:height="match_parent"
    ohos:width="match_parent"
    ohos:orientation="vertical"
    ohos:background_element="$graphic:background_ability_page"
    >
```

绘制应用名称“关键词搜索图片”

```
<Text
    ohos:id="$+id:text_helloworld"
    ohos:height="match_content"
    ohos:width="match_content"
    ohos:background_element="$graphic:background_ability_main"
    ohos:layout_alignment="horizontal_center"
    ohos:text="关键词搜索图片"
```

```
    ohos:text_size="30fp"
    ohos:top_margin="5vp"
    />
```

绘制“图片列表”标题

```
<Text
    ohos:id="$+id:picture_list"
    ohos:height="match_content"
    ohos:width="match_content"
    ohos:background_element="$graphic:background_ability_main"
    ohos:layout_alignment="horizontal_center"
    ohos:text="图片列表"
    ohos:text_size="20fp"
    ohos:top_margin="15vp"
    />
```

创建图片资源列表类

```
<ListContainer
    ohos:id="$+id:picture_list_show"
    ohos:height="200vp"
```

```
    ohos:width="match_parent"
    ohos:orientation="horizontal"
    ohos:left_margin="5vp"
    ohos:right_margin="5vp"
    />
```

绘制“请输入关键词”标题

```
<Text
    ohos:id="$+id:word_seg_title"
    ohos:height="match_content"
    ohos:width="match_content"
    ohos:background_element="$graphic:background_ability_main"
    ohos:left_margin="5vp"
    ohos:text="请输入关键词："
    ohos:text_size="25fp"
    ohos:top_margin="10vp"
    />
```

绘制文本输入窗口

```
<TextField
```

```
    ohos:id="$+id:word_seg_text"
    ohos:height="match_content"
    ohos:width="match_parent"
    ohos:background_element="$graphic:background_ability_main"
    ohos:hint="Enter a statement."
    ohos:left_padding="5vp"
    ohos:right_padding="5vp"
    ohos:text_alignment="vertical_center"
    ohos:text_size="20fp"
    ohos:top_margin="5vp"/>
```

绘制"开始通用文字识别"按钮

```
<Button
    ohos:id="$+id:button_search"
    ohos:width="match_content"
    ohos:height="match_content"
    ohos:text_size="20fp"
    ohos:text="开始通用文字识别"
    ohos:layout_alignment="horizontal_center"
    ohos:top_margin="10vp"
    ohos:top_padding="1vp"
```

```
        ohos:bottom_padding="1vp"
        ohos:right_padding="20vp"
        ohos:left_padding="20vp"
        ohos:text_color="white"
        ohos:background_element="$graphic:button_element"
        ohos:center_in_parent="true"
        ohos:align_parent_bottom="true"
        ohos:bottom_margin="5vp"/>
```

绘制“搜索结果”标题

```
<Text
        ohos:id="$+id:picture_list_result"
        ohos:height="match_content"
        ohos:width="match_content"
        ohos:background_element="$graphic:background_ability_main"
        ohos:layout_alignment="horizontal_center"
        ohos:text="搜索结果"
        ohos:text_size="20fp"
        ohos:top_margin="5vp"
        />
```

创建“搜索结果”图片列表类

```
<ListContainer
    ohos:id="$+id:picture_list_match"
    ohos:height="200vp"
    ohos:width="match_parent"
    ohos:orientation="horizontal"
    ohos:left_margin="5vp"
    ohos:right_margin="5vp"
    />
```

Step 2 ：接下来，我们在“resources>base>layout”目录下的 item_image_layout.xml 文件，并通过以下代码设置图片水平居中（horizontal_center）展示：

```
<DirectionalLayout xmlns:ohos="http://schemas.huawei.com/res/ohos"
                   ohos:height="200vp"
                   ohos:width="205vp">
    <Image
        ohos:id="$+id:select_picture_list"
        ohos:height="200vp"
        ohos:width="200vp"
        ohos:layout_alignment="horizontal_center"
```

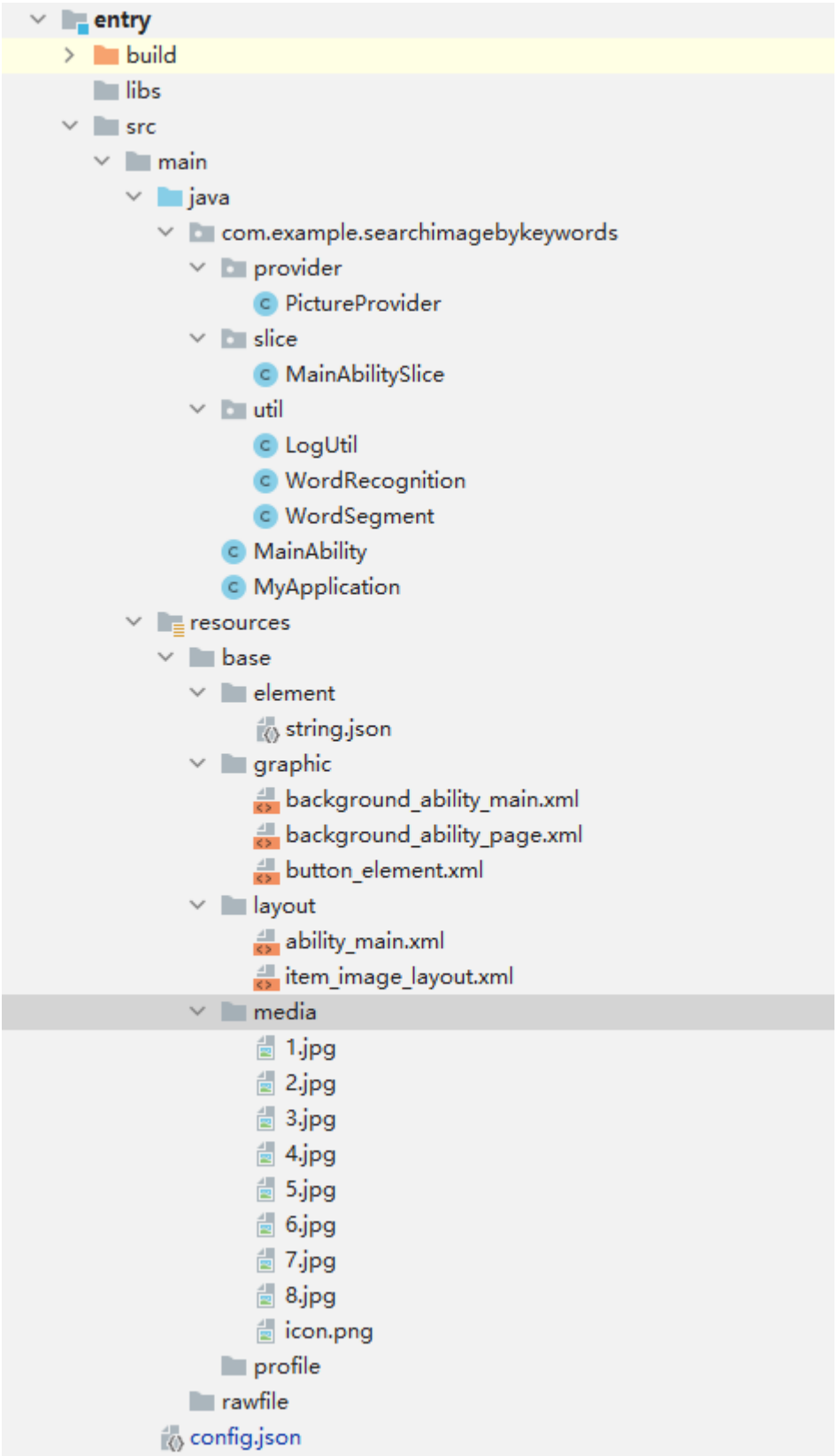
```
        ohos:top_margin="1vp"
        ohos:scale_mode="stretch"
        />
</DirectionalLayout>
```

Step 3 ：我们在“resources>base>graphic”目录下的 background_ability_page.xml 文件中，设置界面背景颜色，代码如下：

```
<shapexmlns:ohos="http://schemas.huawei.com/res/ohos"
        ohos:shape="rectangle">
    <solid
        ohos:color="#FFFAF0"/>
</shape>
</DirectionalLayout>
```

添加图片并展示

Step 1 ：目前应用支持处理的图片格式包括 JPEG、JPG、PNG、GIF、BMP，我们在“resources>base>media”目录下添加 8 张上述任一格式的图片（示例为 jpg 格式，分别命名为 1-8.jpg），且图片须带有文字（当图片无文字，系统会识别返回空字符串），以便运行时有图片数据做测试；



来源：HarmonyOS 开发者微信号 https://mp.weixin.qq.com/s/Oi06hTRUFhlK2PdftE1S7g

Step 2 ：接下来，我们在“java>com.example.searchImagebykeywords>slice”目录下的 MainAbilitySlice 加载图片 id 数组，代码如下：

（说明：示例中目录名称“com.example.searchImagebykeywords”会随创建项目的名称不同而改变，具体以实际情况为准）

```
private int[] pictureLists = new int[]{ResourceTable.Media_1,
ResourceTable.Media_2,
    ResourceTable.Media_3, ResourceTable.Media_4, ResourceTable.Media_5,
    ResourceTable.Media_6, ResourceTable.Media_7, ResourceTable.Media_8};
```

Step 3 ：在“java>com.example.searchImagebykeywords”目录下新建 Package 包并将其命名为“provider”，在 provider 下创建 PictureProvider 文件，其中获取图片 id 数组和 MainAbilitySlice 对象，实现代码如下：

```
public PictureProvider(int[] pictureLists,Context context) {
        this.pictureLists = pictureLists;
        this.context = context;
```

Step 4：在 PictureProvider 文件中，通过以下代码，将图片展示到页面上。

```
@Override
public Component getComponent(int var1, Component var2,
```

```
ComponentContainervar3) {
    ViewHolder viewHolder = null;
    Component component = var2;
    if (component == null) {
        component
=LayoutScatter.getInstance(context).parse(ResourceTable.Layout_item_image_layo
ut,
                    null, false);
        viewHolder = new ViewHolder();
        Component                                                    componentImage
=component.findComponentById(ResourceTable.Id_select_picture_list);
        if (componentImage instanceofImage) {
            viewHolder.image = (Image)componentImage;
        }
        component.setTag(viewHolder);
    } else {
        if (component.getTag() instanceofViewHolder) {
            viewHolder = (ViewHolder)component.getTag();
        }
    }
    if (viewHolder != null) {
       viewHolder.image.setPixelMap(pictureLists[var1]);
```

```
        }
        return component;
    }
```

Step 5：同样，我们在 PictureProvider 文件中，定义 ViewHolder 类，实现以列表的形式展示图片，代码如下：

```
private static class ViewHolder {
Image image;
}
```

识别图片中的文字

Step 1 ：我们在“java>com.example.searchImagebykeywords”目录下新建 Package 包并将其命名为“util”，在 util 下创建 WordRecognition 文件，并通过以下代码，调用文字识别方法对图片文字进行识别。

通用文字识别的核心技术 OCR 可通过拍照、扫描等光学输入方式，把各种票据、卡证、表格、报刊、书籍等印刷品文字转化为图像信息，再利用文字识别技术将图像信息转化为计算机等设备可以使用的字符信息，如果您想了解更多详情，请参考官网相关描述：

通用文字识别概述：

https://developer.harmonyos.com/cn/docs/documentation/doc-guides/ai-text-recognition-overview-0000000000042007

```
public void wordRecognition(Contextcontext, int resId, MainAbilitySlice.MyEventHandle myEventHandle) {
    mediaId = resId;
    // 实例化 ITextDetector 接口
    textDetector =VisionManager.getTextDetector(context);

    // 实例化 VisionImage 对象 image，并传入待检测图片 pixelMap
    pixelMap = getPixelMap(resId);
    VisionImage image =VisionImage.fromPixelMap(pixelMap);

    // 定义 VisionCallback<Text>回调，异步模式下用到
    VisionCallback<Text>visionCallback = getVisionCallback();

    // 定义 ConnectionCallback 回调，实现连接能力引擎成功与否后的操作
    ConnectionCallback connectionCallback= getConnectionCallback(image, visionCallback);

    // 建立与能力引擎的连接
    VisionManager.init(context,connectionCallback);
```

```
}
```

Step 2 ：接着，我们在 WordRecognition 文件中，用异步模式下回调方法，将图片中文字识别结果通过 sendResult()方法发送到主线程，代码如下：

```
private VisionCallback getVisionCallback(){
        return newVisionCallback<Text>() {
            @Override
            public void onResult(Text text){
                sendResult(text.getValue());
            }
        };
 }
```

我们需要了解一下，同步模式是派发任务并在当前线程等待任务执行完成，在返回前，当前线程会被阻塞；异步模式是派发任务并立即返回，返回值是一个可用于取消任务的接口。如果您想了解更多详情，请参考官网相关描述：

线程管理开发指导：

https://developer.harmonyos.com/cn/docs/documentation/doc-guides/thread-mgmt-guidelines-0000000000032130

Step 3 ：同样，在 WordRecognition 文件中，当连接引擎成功后进行文字识别，并将识

别结果通过 sendResult()方法发送到主线程，代码如下：

```
Private ConnectionCallbackgetConnectionCallback(VisionImage image,
VisionCallback<Text>visionCallback) {
    return new ConnectionCallback() {
        @Override
        public void onServiceConnect() {
            // 实例化 Text 对象 text
            Text text = new Text();
            // 通过 TextConfiguration 配置 textDetector()方法的运行参数
            TextConfiguration.Builderbuilder = new TextConfiguration.Builder();
           builder.setProcessMode(VisionConfiguration.MODE_IN);

builder.setDetectType(TextDetectType.TYPE_TEXT_DETECT_FOCUS_SHOOT);
// 此处变量名将会被调整
            builder.setLanguage(TextConfiguration.AUTO);
            TextConfiguration config =builder.build();
           textDetector.setVisionConfiguration(config);
            // 调用 ITextDetector 的 detect()方法
            if (!IS_ASYNC) {
                int result2 =textDetector.detect(image, text, null); // 同步
               sendResult(text.getValue());
```

```
            } else {
                int result2 =textDetector.detect(image, null, visionCallback); // 异
步
            }
        }

        @Override
        public voidonServiceDisconnect() {
            // 释放 成功：同步结果码为 0，异步结果码为 700
            if ((!IS_ASYNC &&(result == 0)) || (IS_ASYNC && (result ==
IS_ASYNC_CODE))) {
                textDetector.release();
            }
            if (pixelMap != null) {
                pixelMap.release();
                pixelMap = null;
            }
            VisionManager.destroy();
        }
    };
 }
```

Step 4 ：在 WordRecognition 文件中，将文字识别结果发送到主线程（MainAbilitySlice 类中接收），代码如下：

```
public void sendResult(String value) {
    if (textDetector != null) {
        textDetector.release();
    }
    if (pixelMap != null) {
        pixelMap.release();
        pixelMap = null;
        VisionManager.destroy();
    }
    if (value != null) {
        maps.put(mediaId, value);
    }
    if ((maps != null) &&(maps.size() == pictureLists.length)) {
        InnerEvent event =InnerEvent.get(1, 0, maps);
        handle.sendEvent(event);
    } else {
        wordRecognition(slice,pictureLists[index], handle);
        index++;
    }
```

}

提取用户输入的关键词

Step 1：我们在“java>com.example.searchImagebykeywords>util”目录下新建 WordSegment 文件。并在其中获取 MainAbilitySlice 传递的环境参数并进行分词操作，同步方式调用 sendResult()方法将分词结果发送到主线程，代码如下。

我们需要了解，分词作为文本信息提取的第一步，分词模块提供了文本自动分词的接口，对于一段输入文本，可以自动进行分词，同时提供不同的分词粒度。如果您想了解更多详情，请参考官网相关描述：

分词概述：

https://developer.harmonyos.com/cn/docs/documentation/doc-guides/ai-word-segmentation-overview-0000001051092452

```
public void wordSegment(Context context,String requestData, MainAbilitySlice.MyEventHandle myEventHandle) {
        slice = context; //MainAbilitySlice.this
        handle = myEventHandle; //MyEventHandle 对象

        // 使用 NluClient 静态类进行初始化，通过异步方式获取服务的连接。
```

```
        NluClient.getInstance().init(context,new OnResultListener<Integer>() {
                @Override
                public voidonResult(Integer resultCode) {
                        if (!IS_ASYNC){
                                // 分词同步方法
                                ResponseResultresponseResult =
NluClient.getInstance().getWordSegment(requestData,

NluRequestType.REQUEST_TYPE_LOCAL);

sendResult(responseResult.getResponseResult());
                                release();
                        } else {
                                // 分词异步方法
                                wordSegmentAsync(requestData);
                        }
                }
        }, true);
}
```

请注意这里分词文本限制在 500 字以内，目前只支持中文语境，支持的语言有：中文、英文、日语、韩语、俄语、意大利语、西班牙语、葡萄牙语、德语，以及法语（将来会增加更

多语种），但不支持手写字体识别。

Step 2：在 WordSegment 文件中，异步请求回调此方法，通过 sendResult()方法将分词结果发送到主线程，代码如下：

```
private void wordSegmentAsync(StringrequestData) {
        ResponseResult responseResult
=NluClient.getInstance().getWordSegment(requestData,
                NluRequestType.REQUEST_TYPE_LOCAL,new
OnResultListener<ResponseResult>() {
                    @Override
                    publicvoid onResult(ResponseResult
asyncResult) {

sendResult(asyncResult.getResponseResult());
                        release();
                    }
                });
}
```

Step 3：在 WordSegment 文件中，将分词结果发送到主线程中（MainAbilitySlice 类中接收），代码如下：

```
private void sendResult(String result) {
        List lists = null;// 分词识别结果
        // 将 result 中分词结果转换成 list
        if(result.contains("\"message\":\"success\"")) {
                String words =result.substring(result.indexOf(WORDS) + STEP,
                                result.lastIndexOf("]")).replaceAll("\"","");
                if ((words == null) ||("".equals(words))) {
                        lists = newArrayList(1);// 未识别到分词结果，返回"no
keywords"
                        lists.add("nokeywords");
                } else {
                        lists =Arrays.asList(words.split(","));
                }
        }

        InnerEvent event =InnerEvent.get(TWO, ZERO, lists);
        handle.sendEvent(event);
}
```

根据关键词匹配图片

Step 1：在“java>com.example.searchImagebykeywords>slice”目录下的MainAbilitySlice 文件中，根据关键词匹配待识别的图片，并将其展示，为了方便对图片中提取出的文字进行关键词识别，这里将会把字符串类型转换为列表类型，代码如下：

```
private void matchImage(List<String>list) {
        Set<Integer> matchSets =new HashSet<>();
        for (String str: list) { // 遍历分词结果
                // imageInfos 待识别图片通用文字识别结果
                for (Integer key :imageInfos.keySet()) {
                        if(imageInfos.get(key).indexOf(str) != NEG_ONE) {
                                matchSets.add(key);
                        }
                }
        }
        // 获得匹配的图片
        matchPictures = newint[matchSets.size()];
        int i = 0;
        for (int match: matchSets) {
                matchPictures[i] =match;
                i++;
```

```
        }
        // 展示图片
        setSelectPicture(matchPictures,LIST_CONTAINER_ID_MATCH);
}
```

Step 2：同样，我们在 MainAbilitySlice 文件中，进一步将“匹配图片”投放到到页面上，代码如下：

```
private void setSelectPicture(int[]pictures, int id) {
        // 获取图片
        PictureProvider newsTypeAdapter =new PictureProvider(pictures, this);

        Component componentById =findComponentById(id);
        if (componentById instanceofListContainer) {
                ListContainerlistContainer = (ListContainer) componentById;
                listContainer.setItemProvider(newsTypeAdapter);
        }
}
```

日志打印

Step 1：我们在“java>com.example.searchImagebykeywords>util”目录下创建LogUtil.java 类对日志打印进行了封装，实现代码如下：

```
package com.huawei.searchimagebykeywords.util;

import ohos.hiviewdfx.HiLog;

import ohos.hiviewdfx.HiLogLabel;
/**
 * logUtil util
 *
 * @since 2021-01-15
 */

public class LogUtil {
    private static final String TAG_LOG = "LogUtil";
    private static final HiLogLabel LABEL_LOG = new HiLogLabel(0, 0,
LogUtil.TAG_LOG);
    private static final String LOG_FORMAT = "%{public}s: %{public}s";
    private LogUtil() {
    }
    /**
```

```
     * Print info log
     *
     * @param tag log tag
     * @param msg log message
     */
    public static void info(String tag, String msg) {
        HiLog.info(LABEL_LOG, LOG_FORMAT, tag, msg);
    }
    /**
     * Print info log
     *
     * @param tag log tag
     * @param msg log message
     */
    public static void error(String tag, String msg) {
        HiLog.info(LABEL_LOG, LOG_FORMAT, tag, msg);
    }
}
```

Step 2：我们在“java>com.example.searchImagebykeywords>util”目录下的WordRecognition 文件中，调用 LogUtil.error 打印 error 级别的日志，实现代码如下：

当获取图片资源异常时，日志打印异常提示

```
catch (IOException | NotExistException e) {

    LogUtil.error("get pixelmap failed, read resource bytes failed, ",
e.getLocalizedMessage());
```

当读取图片类异常时，日志打印异常提示

```
catch (IOException e) {
    LogUtil.error("OrcAbilitySlice.getPixelMap", "read resource failed ")
```

当关闭读取图片类异常时，日志打印异常提示

```
catch (IOException e) {
    LogUtil.error("OrcAbilitySlice.getPixelMap", "close output failed");
```

点击 Run > Run ‘entry'运行 hap 包，我们在搜索框中输入需要分词的关键词，点击【开始通用文字识别】按钮进行关键词搜索图片，在“搜索结果”下方就可以看到包含关键词的图片啦！