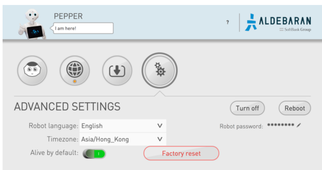
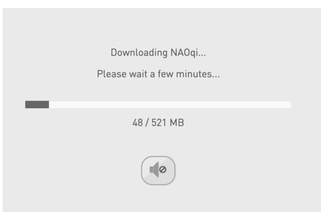
Using Xunfei Semantic Understanding on Pepper

Preparation

For accessing third-party semantic understanding (nlp), this article takes Xunfei as an example to start the explanation. Firstly, enter the Xunfei open platform (https://aiui.iflyos.cn/apps), as shown in Figure 1.1 below.



In the above figure 1.1 did not create an application click on 'Create Application' to enter the application creation interface, as shown in Figure 1.2 below:



As shown in Figure 1.2 above, firstly, enter the application name, select Android as the application platform, and select the corresponding category according to your usage. Finally, click 'OK to create' to create the app.

After you have created the app, click My Apps to enter the app details page, as shown in Figure 1.3 below:

In Figure 1.3 above, the application details page in the application information can be seen inside the APPID and KEY we need to pack in the client; in the application configuration column corresponding to the application configuration page including speech recognition, semantic understanding and speech synthesis of the three parts of the demo is mainly used in the application of semantic understanding, so the semantic understanding must be turned on; if the Xunfei speech synthesis, the speech synthesis switch should also be open.

In Figure 1.3, click on the right side of the 'development tools' to enter the download page, click on 'Download the latest version of AIUI SDK' to download the SDK, as shown in Figure 1.4 below:

Figure 1.4 shows the following:

As shown in Figure 1.4 above download the SDK after selecting the AI skills you need.

Project Configuration

1) Create a new project Android project.

2) Import SDK:

Refer to Xunfei import SDK address as below:

https://doc.iflyos.cn/aiui/sdk/mobile_doc/quick_start.html#%E5%BC%80%E5%8F%91%E6%AD%A5%E9%AA%A4

Note: aiui_phone file with demo in the configuration of aiui_phone, there is no use of the configuration parameters 'iat' and 'audioparams' corresponding to the object to remove; close the vad in the 'vad_enable': '0', other configurations to remove; remove all the comments; replace 'appid' and 'appKey' in 'login' with your own 'appid' and 'appKey' of the application you registered on Xunfei's open platform.

Project Integration

1) Based on Xunfei's nlp and tts's pepper's voice interaction process, the code implementation is shown in figure 3.1 below:

```
public class MainChatbot extends BaseChatbot {

    private static final String TAG = "MainChatbot";
    private QiContext qiContent;
    private Context mContext;

    public MainChatbot(QiContext qiContent, Context context) {
        super(qiContent);
        this.qiContent = qiContent;
        mContext = context;
    }

    @Override
    public StandardReplyReaction replyTo(Phrase phrase, Locale locale) {
        if (phrase != null) {
            String text = phrase.getText();
            if (text.isEmpty()) {
                Log.d( tag: "MyChatbotReaction",  msg: "answer empty: ");
                EmptyChatbotReaction emptyReac = new EmptyChatbotReaction(qiContent, "没有听到你说啥",  type: 0);
                return new StandardReplyReaction(emptyReac, ReplyPriority.FALLBACK);
            } else {
                Log.d( tag: "MyChatbotReaction",  msg: "nuance could asr string is :" + text);
                IFlytekNlpReaction iFlytekNlpReaction = new IFlytekNlpReaction(qiContent, text);
                return new StandardReplyReaction(iFlytekNlpReaction, ReplyPriority.NORMAL);
            }
        } else {
            Log.d( tag: "MyChatbotReaction",  msg: "answer empty1: ");
            EmptyChatbotReaction emptyReac = new EmptyChatbotReaction(qiContent, "没有听到你说啥",  type: 3);
            return new StandardReplyReaction(emptyReac, ReplyPriority.FALLBACK);
        }
    }

}
```

In Figure 3.1, StandardReplyReaction replyTo(Phrase phrase, Locale locale)

return parameter phrase for pepper will hear the sound into the text. IFlytekNlpReaction iFlytekNlpReaction = **new** IFlytekNlpReaction(**qiContent**, text); is to call up the code of Xunfei's NLP StandardReplyReaction(iFlytekNlpReaction, ReplyPriority.**NORMAL**) is the code to crank up the behaviour of the pepper listening. This forms a complete voice interaction process.

Note: If you are not familiar with some of the technical concepts in voice interaction, you can refer to the link:

https://blog.csdn.net/ZLJ925/article/details/79016180

2) Xunfei nlp code access details:

We encapsulate Xunfei initialisation and nlp in a class com.softbankrobotics.nlp.chat. IFlytekNlpReaction.

a) Initialisation part of the code:

```
public AIUIAgent createAgent(MyAIUIListener mAIUIListener) {
    Log.i(TAG,  msg: "create aiui agent");
    try {
        String params = IOUtils.fromAsset(mContext,  assetName: "cfg/aiui_phone.cfg");
        params = params.replace( target: "\n",  replacement: "").replace( target: "\t",  replacement: "").replace( target: " ",  replacement: "");
        AIUIAgent aIUIAgent = AIUIAgent.createAgent(mContext, params, mAIUIListener);
        return aIUIAgent;
    } catch (Exception e) {
        Log.e(TAG, e.getMessage());
        return null;
    }
}
```

b) The code for the semantic understanding part of Cybernetics:

```
private void sendNlpMessage(AIUIAgent mAIUIAgent) {
    if (AIUIConstant.STATE_WORKING != mAIUIState) {
        AIUIMessage wakeupMsg = new AIUIMessage(AIUIConstant.CMD_WAKEUP,  i1: 0,  i2: 0,  s: "",  bytes: null);
        mAIUIAgent.sendMessage(wakeupMsg);
    }

    Log.i(TAG,  msg: "start text nlp");
    try {
        String params = "data_type=text,tag=text-tag";
        byte[] textData = question.getBytes( charsetName: "utf-8");

        AIUIMessage write = new AIUIMessage(AIUIConstant.CMD_WRITE,  i1: 0,  i2: 0, params, textData);
        mAIUIAgent.sendMessage(write);


    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
}
```

Note: Cyberoam's tts are encapsulated in the bottom layer, nlp will automatically call up Cyberoam's tts and play the sound after processing.

c) Asynchronous task processing:

In order to solve the problem of self-questioning, the solution is that you can't listen when you speak. Because Xunfei nlp event processing are asynchronous, you need to turn asynchronous to synchronous processing, finished talking in order to let the pepper to listen, as follows Figure 3.4.

d) Implement AIUIListener interface can get various parameters of Xunfei event callback.

```
    /
    @Override
    public void runWith(SpeechEngine speechEngine) {
        CountDownLatch countDownLatch = new CountDownLatch(1);
        MyAIUIListener myAIUIListener = new MyAIUIListener();
        myAIUIListener.setCountDownLatch(countDownLatch);
        AIUIAgent aiuiAgent = createAgent(myAIUIListener);

        sendNlpMessage(aiuiAgent);
        try {
            countDownLatch.await();//需要注意超时问题
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        if (!useIFlytekTTS) {
            // Build the Say action
            Say say = SayBuilder.with(speechEngine)
                    .withText(myAIUIListener.getAnswer())
                    .build();

            com.aldebaran.qi.Future fSay = say.async().run();

            try {
                fSay.get();
            } catch (ExecutionException e) {
                Log.e(TAG,  msg: "Error during Say", e);
            } catch (CancellationException e) {
                Log.i(TAG,  msg: "Interruption during Say");
            }
        }
        aiuiAgent.destroy();
    }
```

As in Figure 3.4, CountDownLatch is used to solve the problem of asynchronous to synchronous, in the code countDownLatch.await() to block the current thread, and wait for Xunfei's semantic comprehension to be executed to execute Xunfei's speech synthesis or call Qisdk's Say method to speak the returned text.

Full code example

Developers can visit SoftBank Robotics China Github to see the full code example of this chapter.