Eva Android SDK

Version 2.0

Introduction

Evature develops free text understanding of travel related queries.
It includes a “Dialog Engine” which chats with the user and integrates within your app - triggering your travel search with the search criteria specified by the user.

With the *new* Eva SDK you can **integrate a complete voice-based search into your app in just a few minutes**!

Step 1: Setup the SDK in your IDE

First download or clone the SDK - the code repository is located at
https://github.com/evature/android

Importing to Android Studio (recommended)

1. Import the SDK project to Android Studio:
   a. Choose from the menu: File > New > Import Project
   b. Choose the build.gradle file in the *evasdk* folder

2. In your project's *build.grade* add to the dependencies:

```
dependencies {
   /* your other depenedcies here */
   releaseCompile project(path: ':evasdk', configuration: 'release')
   debugCompile project(path: ':evasdk', configuration: 'debug')
}
```

Importing to an Eclipse Workspace (deprecated)

Integrating the SDK into an Eclipse project is more complex because it requires modifying the AndroidManifest.xml and copying many resource files:

1. Create a new Eclipse Android project, from existing sources and choose the SDK folder
2. Mark the project as a library in the project settings "Android Tab"
3. Add a project dependency on Android Support library v4
4. Copy the permissions and activities entries from the SDK's Android Manifest file to your project's Manifest file
5. Copy all the resource files (files under the *res* folder) from the SDK to your *res* folder (keeping the sub-directories structure).

Note if/when the SDK will be updated, you may have to update the Android Manifest entries and/or the resource files that were copied from the SDK.

Step 2: Initialize and Configure Eva

The minimal required setup is to call `AppSetup.initEva` with three parameters:

1. Your API_KEY - received when you register to Evature web service
2. Your SITE_CODE - received with your API_KEY
3. Your App handler class, see more details later at step 4.

For example:

```
public class MyTravelApp extends Application {
   @Override
   public void onCreate() {
       super.onCreate();
       AppSetup.setScope(AppScope.Flight, AppScope.Hotel);
       AppSetup.evaLogs(BuildConfig.DEBUG);  // enable Eva logs on Debug builds
       AppSetup.initEva(API_KEY, SITE_CODE, new MyTravelAppHandler());
   }
}
```

Optional settings can be accessed by public static fields/methods of the AppSetup class. They include:

1. App Scope - this lets Eva know what are the application capabilities (eg. flight search, hotel search, car search, etc..)
2. Enable/Disable Eva logs.
3. autoOpenMicrophone - set to True to start recording automatically when Eva Chat expects a user to talk.

Step 3: Add a button to trigger the Chat Screen

In activities where you wish to trigger a chat screen you can add a default Microphone button (recommended):

```
public class SearchScreenActivity extends FragmentActivity {
  @Override
  public void onCreate ( Bundle savedInstanceState )
  {
     super.onCreate(savedInstanceState);
//   Your activity's onCreate code eg.
//   setContentView(R.layout.search_screen_layout);

     //   Add a default "Microphone" button
     EvaChatTrigger.addDefaultButton(this);
  }
```

Or alternatively, you can add your own button and trigger the chat screen when it is clicked:

```
public void onCreate ( Bundle savedInstanceState )
{
     super.onCreate(savedInstanceState);
//   Your activity's onCreate code eg.
//   setContentView(R.layout.search_screen_layout);

     Button btn = ((Button) findViewById(R.id.my_chat_btn));
     btn.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
           // Show the chat screen:
           EvaChatTrigger.startSearchByVoice(FragmentActivity.this);
        }
```

```
    });
```

Note:
Eva assumes the host activity is of type *android.support.v4.app.FragmentActivity*. If this isn't the case in your application, let us know and we can provide a workaround.

Step 4: Implement your App callbacks

Remember the "`MyTravelAppHandler`" which we passed to *`initEva`* in step one? This object should implement interfaces from `com.evature.evasdk.appinterface` namespace.

These interfaces include:

- CarCount
- CarSearch
- CruiseCount
- CruiseSearch
- FlightCount - count how many flights match the criteria requested by the user.
- FlightSearch - trigger a search for the flights using the criteria requested by the user.
- HotelCount
- HotelSearch

In the future we will add more interfaces, stay tuned!

Of course, you don't have to implement all the interfaces, only those that your app supports!

Eva will automatically infer which features should be enabled based on which interfaces your app handler implements.

For example, if your app only supports searching for flights then you only need to implement ony interface: ***FlightSearch***, which (currently) includes two methods: *handleOneWayFlightSearch*, *handleRoundTripFlightSearch.*

As you can tell from the above list, the current interfaces are pairs of *Search and *Count, see below their different meaning.
Normally you should implement the *Search interface, and the *Count interface is optional (but highly recommended).

App Search interfaces

The methods in these interfaces receive all the many search criteria as function arguments.

All you have to do is translate the arguments to your own types and trigger your own search function and you are done!

The different interfaces and their many parameters are documented in the javadocs. They are pretty straightforward and simply describe the criteria as entered by the user (eg. what airline he wants, on what date to depart, etc…). One special argument is different, it is called "isComplete" and is described below.

Note: null values of parameters -
If the user did not request a certain search criteria, it will be passed as "null", so please remember to check that each argument isn't null before you use it.
For example the "nonStop" parameter will be *True* if the user requested non-stop flights, *False* if the user requested NOT non-stop flights, and *null* if the user did not mention this criteria at all.

IsComplete

One of the search methods' parameters is called "*isComplete*". Unlike the other parameters, this is not a search criteria requested by the user but instead it is Eva's dialog state.
While the user has not entered **all** the mandatory search parameters Eva will ask the user for more input and will trigger a search with the parameters entered so far and *isComplete=False*.
After the user has entered **all** the mandatory search parameters Eva will trigger a search with *isComplete=True*, and Eva will end the dialog by saying "searching for <*text describing the search*>".

The most common use case would be ignoring calls with *isComplete=False*, and trigger a search only when *isComplete=True,* and that search would open a new screen (ie. new activity or fragment).

App Count interfaces

If you app supports counting how many search results will be for a partial search (ie. isComplete=False), it is highly recommended that you implement the matching **Count* interface.

The **Count* interfaces (eg. HotelCount, FlightCount, etc..) receive a the same set of arguments as their matching **Search* interfaces, except that instead of the "*sortBy*" and "*sortOrder*" arguments they receive a callback of the type `AsyncCountResult`.
Simply activate your counting logic (it can be easily an async function, eg. request to server or DB) and call the *handleCountResult* method of the callback with your count results.

When implemented, Eva will use these count results to show below Eva's text while the user chats.

Advanced Integration

Look & Feel Customizations

Simple customizations of the Eva chat screen and the default microphone button are very easy; all the strings, colors, layouts and bitmaps are in resource files.

Simply copy the resources to your project and modify them to fit your needs. All the resource file names and IDs are prefixed with "evature_" so there should be no conflict with your files/IDs.

Let us know if you are encountering problems or if you wish to customize features which can't be done by modifying the resources. For example, changing the "chat bubbles" look (aside from color, padding, etc.) may require more some code modifications.

Google Now Integration

By integrating with Eva's SDK you get a "free" integration with Google Now!

Users will be able to say to their phone:

"OK Google. Search for New York to Tokyo on *MyTravelApp*"

And this will open your app and start a chat with Eva searching for "New York to Tokyo" !

Unfortunately, the updated app needs to be published in Google Play to fully test the Google Now integration, but you can try it manually with ADB -

Run from your command line:

```
adb shell am start -a com.google.android.gms.actions.SEARCH_ACTION -e query
"New York to Tokyo" your.app.package.here
```