天津大学本科生实验报告专用纸

学院 智算学部 年级 17 专业 软件工程 班级 1 姓名 刘坤鑫 学号 3017218061

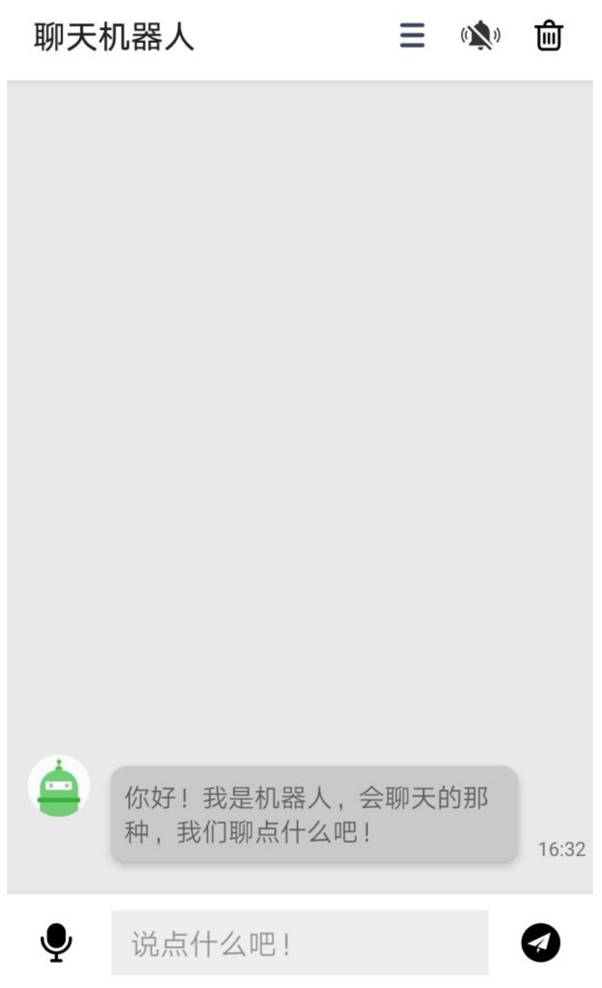
课程名称 移动平台开发 实验日期 2019.5.23 成绩_________

同组实验者____________________

基于语音识别的智能聊天机器人

一、 功能概述

1. 基于科大讯飞，实现语音识别，语音合成，支持方言、多种发声。
2. 基于图灵机器人，实现智能聊天，包含以下功能：数字计算、语料库、中英互译、聊天对话等。详见 https://www.kancloud.cn/turing/www-tuling123-com/718219。
3. 仿微信聊天界面，简洁风格，丝般顺滑。



天津大学本科生实验报告专用纸

二、 具体实现

2.1 主界面设计

界面分为三个部分：工具栏、对话栏、输入栏。

工具栏包括两个部分：标题、选项设置。选项设置根据需求需要前后添加了三个 button，分别实现以下功能：清空聊天记录、语音播报开关、发声人设置。

具体代码如下：

```
<android.support.design.widget.AppBarLayout
    android:id="@+id/abl"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/colorPrimary"
    app:layout_constraintTop_toTopOf="parent">

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <android.support.v7.widget.Toolbar
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:layout_constraintTop_toTopOf="parent"

            app:title="聊天机器人"

            tools:layout_editor_absoluteX="0dp" />

        <ImageView
            android:id="@+id/iv_delete"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:padding="5dp"
            android:layout_marginEnd="12dp"
            android:src="@drawable/shanchu"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <ImageView
            android:id="@+id/iv_sound"
            android:layout_width="30dp"
            android:layout_height="30dp"
```

```
                android:padding="3dp"
                android:layout_marginEnd="12dp"
                android:src="@drawable/jinyin"
                android:background="?attr/selectableItemBackground"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toStartOf="@id/iv_delete"
                app:layout_constraintTop_toTopOf="parent" />

            <ImageView
                android:id="@+id/iv_menu"
                android:layout_width="30dp"
                android:layout_height="30dp"
                android:padding="5dp"
                android:layout_marginEnd="12dp"
                android:src="@drawable/caidan"
                android:background="?attr/selectableItemBackground"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toStartOf="@id/iv_sound"
                app:layout_constraintTop_toTopOf="parent" />
        </android.support.constraint.ConstraintLayout>
    </android.support.design.widget.AppBarLayout>
```

其中 button 控件采用了 ImageView 控件实现。

对话框要求实现仿微信聊天界面的功能。

分析聊天界面，有以下要素：人物头像、对话框、发送时间。其中人物对话己方在右，对方在左。

采用的解决方案如下：先实现我方和对方的 item 元素：

view_message_robot_item.xml：

```
    <ImageView
        android:id="@+id/iv_avatar"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:elevation="4dp"
        android:layout_marginStart="12dp"
        android:src="@drawable/robot_avatar"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tv_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="12dp"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp"
        android:background="@drawable/bg2"
        android:elevation="4dp"
        android:maxWidth="250dp"
        android:minHeight="40dp"
        android:padding="8dp"
        android:text="1"
        android:textSize="16sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@id/iv_avatar"
        app:layout_constraintTop_toTopOf="@id/iv_avatar" />

    <TextView
        android:id="@+id/tv_date"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="12dp"
        android:text="11:11"
        android:textSize="12sp"
        app:layout_constraintBottom_toBottomOf="@id/tv_text"
        app:layout_constraintStart_toEndOf="@id/tv_text" />
```

分别表示头像、对话框、时间元素。其中 android:elevation="4dp"实现了阴影效果，在 android:background="@drawable/bg2"里则实现了边框圆角的效果。

bg2.xml：

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="#CAC8C8" />
    <corners android:topLeftRadius="10dp"
        android:topRightRadius="10dp"
        android:bottomRightRadius="10dp"
        android:bottomLeftRadius="10dp"/>
</shape>
```

我方对话框实现完全同理。

实现 item 后，外层采用 RecyclerView 线性排布。

输入栏包含三个元素：语音识别 button、输入文本框、发送 button。

```
    <android.support.constraint.ConstraintLayout
        android:id="@+id/cl_bottom"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:background="#ffffff"
    android:elevation="8dp"
    app:layout_constraintBottom_toBottomOf="parent">

    <ImageView
        android:id="@+id/iv_voice"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_marginStart="12dp"
        android:padding="8dp"
        android:src="@drawable/yuyin"
        android:background="?attr/selectableItemBackground"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/et_message"
        android:layout_width="0dp"
        android:layout_height="40dp"
        android:layout_marginStart="12dp"
        android:layout_marginEnd="12dp"
        android:background="#11000000"
        android:hint="说点什么吧！"
        android:maxLines="1"
        android:paddingStart="12dp"
        android:paddingEnd="12dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@id/iv_send"
        app:layout_constraintStart_toEndOf="@id/iv_voice"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/bt_voice"
        android:layout_width="0dp"
        android:layout_height="40dp"
        android:layout_marginStart="12dp"
        android:layout_marginEnd="12dp"
        android:background="#ffffff"
        android:gravity="center"
        android:maxLines="1"
        android:paddingStart="12dp"
        android:paddingEnd="12dp"
        android:text="点击开始说话"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@id/iv_send"
        app:layout_constraintStart_toEndOf="@id/iv_voice"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/iv_send"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_marginEnd="12dp"
        android:padding="8dp"
        android:background="?attr/selectableItemBackground"
        android:src="@drawable/fasong"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

2.2 设置界面

此界面用于设置发声人。包括设置离线合成还是在线合成。

本项目期待效果是可以切换离线合成和在线合成，在此基础上再选择发声人。所以需要分两个子界面，一个是离线合成的设置，一个是在线合成的设置。

activity_set_pronunciation.xml：

```
<android.support.design.widget.AppBarLayout
    android:id="@+id/abl"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/colorPrimary"
    app:layout_constraintTop_toTopOf="parent">

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <android.support.v7.widget.Toolbar
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:layout_constraintTop_toTopOf="parent"
```

```
                    app:title="设置发音人" />

            </android.support.constraint.ConstraintLayout>
        </android.support.design.widget.AppBarLayout>

        <RadioGroup
            android:id="@+id/rg"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="24dp"
            android:gravity="center"
            android:orientation="horizontal"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@id/abl">

            <RadioButton
                android:id="@+id/rb_offline"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"

                android:text="离线合成" />


            <RadioButton
                android:id="@+id/rb_online"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="30dp"

                android:text="在线合成" />

        </RadioGroup>

        <View
            android:id="@+id/view"
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:layout_marginTop="24dp"
            android:background="#11000000"
            app:layout_constraintTop_toBottomOf="@id/rg" />

        <android.support.v7.widget.RecyclerView
            android:id="@+id/name_list"
            android:layout_width="match_parent"
@SuppressLint({"CheckResult", "ClickableViewAccessibility", "HandlerLeak"})
```

```
            android:layout_height="0dp"
            android:layout_marginBottom="12dp"
            app:layoutManager="android.support.v7.widget.LinearLayoutManager"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintTop_toBottomOf="@id/view">


        </android.support.v7.widget.RecyclerView>
```

其中两个 radiobutton 切换离线/在线合成，recyclerview 存可选发声人。

发声人另外写了一个 item，由文本和 checkbox 组成：

view_name_item.xml：

```
        <TextView
            android:id="@+id/tv_name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="12dp"
            android:textSize="16sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />



        <CheckBox
            android:id="@+id/cb"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="12dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
```

2.3 后台逻辑

集成了讯飞语音识别和图灵机器人的 api（需要注册并认证），在构建网络请求的时候使用了 Retrofit2 框架，在存储聊天记录的时候用了 Realm。

相关参考：

讯飞 api 文档：https://doc.xfyun.cn/msc_android/index.html

图灵机器人文档：https://www.kancloud.cn/turing/www-tuling123-com/718218

Retrofit2 框架：https://www.jianshu.com/p/f2644cc784f3

https://www.jianshu.com/p/b25669052335

Realm：https://blog.csdn.net/chen_changtui/article/details/83348319

MainActivity.java：

```
public class MainActivity extends AppCompatActivity {

    private MessageAdapter adapter;
    private boolean isPlayMessage = false;
    private RecyclerView recyclerView;

    private Handler handler = new Handler() {
        @Override
        public void handleMessage(android.os.Message msg) {
            super.handleMessage(msg);
            recyclerView.scrollToPosition(adapter.data.size() - 1);
        }
    };

    private Realm realm;

    private MessagePlayProfile profile;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        new RxPermissions(this).request(
                Manifest.permission.RECORD_AUDIO
        ).subscribe(b -> {
            if (!b) MainActivity.this.finish();
        }, Throwable::printStackTrace);


        realm = Realm.getDefaultInstance();
        profile = realm.where(MessagePlayProfile.class).findFirst();

        initMessageList();
        initUI();
    }

    /**
     * 初始化页面相关控件
     */
```

```
    private void initUI() {
        final EditText messageInput = findViewById(R.id.et_message);
        final ImageView sendIv = findViewById(R.id.iv_send);
        final ImageView voiceIv = findViewById(R.id.iv_voice);
        final ImageView deleteIv = findViewById(R.id.iv_delete);
        final ImageView soundIv = findViewById(R.id.iv_sound);
        final ImageView menuIv = findViewById(R.id.iv_menu);

        messageInput.setOnTouchListener((v, event) -> {
            if (event.getAction() == MotionEvent.ACTION_UP) {
                handler.sendEmptyMessageDelayed(0, 100);
            }
            return false;
        });

        //点击后打开讯飞语音识别进行语音识别

        voiceIv.setOnClickListener(v -> {
            RecognizerDialog mDialog = new RecognizerDialog(MainActivity.this, i -> {
            });

            mDialog.setListener(new RecognizerDialogListener() {
                Gson gson = new Gson();
                StringBuilder builder = new StringBuilder();

                @Override
                public void onResult(RecognizerResult recognizerResult, boolean b) {
                    String resultString = recognizerResult.getResultString();

                    //收集文字

                    if (resultString != null) {
                        XFResult result = gson.fromJson(resultString, XFResult.class);
                        builder.append(result.getWord());
                    }

                    //语音结束后将收集到的文字作为消息发出

                    if (b) {
                        sendMessage(builder.toString());
                    }
                }

                @Override
                public void onError(SpeechError speechError) {}
                messageInput.setText("");
```

```java
            });
            mDialog.show();
        });

        //机器人消息的语音播报开关
        soundIv.setOnClickListener(v -> {
            if (isPlayMessage) {
                soundIv.setImageDrawable(getDrawable(R.drawable.jinyin));
            } else {
                soundIv.setImageDrawable(getDrawable(R.drawable.shenyin));
            }
            isPlayMessage = !isPlayMessage;
        });

        //清空所有消息
        deleteIv.setOnClickListener(v -> new AlertDialog.Builder(MainActivity.this)
                .setTitle("删除消息")

                .setMessage("此操作将会删除所有消息")

                .setPositiveButton("删除", (dialog, which) -> {
                    realm.executeTransaction(_realm -> _realm.deleteAll());
                    adapter.clear();
                    dialog.dismiss();
                })
                .setNegativeButton("取消", (dialog, which) -> dialog.dismiss())
                .create().show());

        menuIv.setOnClickListener(v -> {
            startActivity(new Intent(this, SetPronunciationActivity.class));
        });

        //发送文本消息
        sendIv.setOnClickListener(v -> {
            String text = messageInput.getText().toString().trim();
            if (!TextUtils.isEmpty(text)) {
                sendMessage(messageInput.getText().toString().trim());
```

```java
                }
            });
    }

    private void initMessageList() {
        recyclerView = findViewById(R.id.messageList);
        LinearLayoutManager layoutManager = new LinearLayoutManager(this);
        layoutManager.setStackFromEnd(true);
        recyclerView.setLayoutManager(layoutManager);


        RealmResults<Message> data = realm.where(Message.class).sort("date", Sort.ASCENDING).findAll();
        adapter = new MessageAdapter(this, realm.copyFromRealm(data));
        recyclerView.setAdapter(adapter);

        adapter.registerAdapterDataObserver(new RecyclerView.AdapterDataObserver() {

            //出现新消息时滚动到底部，保持列表始终显示最新的消息
            @Override
            public void onItemRangeInserted(int positionStart, int itemCount) {
                super.onItemRangeInserted(positionStart, itemCount);
                recyclerView.scrollToPosition(adapter.data.size() - 1);
            }

        });
        //列表初始化完成后滚动到底部
        recyclerView.scrollToPosition(adapter.data.size() - 1);
    }

    private void sendMessage(String text) {
        //先保存自己发送的消息
        adapter.addMessage(new Message(text, true));


        //构造请求体发送消息
        TulingRequstBody body = new TulingRequstBody();
```

```
            body.perception.inputText.text = text;
            NetHelper.getInstance().sendMessage(body)
                    .observeOn(AndroidSchedulers.mainThread())
                    .subscribe(result -> {
                        TulingResultBody.Results results = result.results.get(0);


                        //获取返回的文本内容并保存消息，然后根据开关选择是否播报消息

                        if (results.resultType.equals("text")) {
                            adapter.addMessage(new Message(results.values.text, false));
                            if (isPlayMessage) {
                                playMessage(results.values.text);
                            }
                        }


                    }, throwable -> Toast.makeText(MainActivity.this, "机器人没能收到这条消息",
Toast.LENGTH_SHORT).show());
    }

    private void playMessage(String text) {
        SpeechSynthesizer speechSynthesizer = SpeechSynthesizer.createSynthesizer(this, null);
        speechSynthesizer.setParameter(SpeechConstant.PARAMS, null);

        //配置在线或离线合成

        if (!profile.isLocal) {
            speechSynthesizer.setParameter(SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_CLOUD);
            speechSynthesizer.setParameter(SpeechConstant.VOICE_NAME, profile.code);

        } else {
            speechSynthesizer.setParameter(SpeechConstant.ENGINE_TYPE, SpeechConstant.TYPE_LOCAL);
            String resourcePath = getResourcePath();
            speechSynthesizer.setParameter(ResourceUtil.TTS_RES_PATH, resourcePath);
            speechSynthesizer.setParameter(SpeechConstant.VOICE_NAME, profile.code);
        }
        speechSynthesizer.setParameter(SpeechConstant.VOICE_NAME, profile.code);
        speechSynthesizer.setParameter(SpeechConstant.SPEED, "50");
        speechSynthesizer.setParameter(SpeechConstant.PITCH, "50");
        speechSynthesizer.setParameter(SpeechConstant.VOLUME, "50");
        speechSynthesizer.setParameter(SpeechConstant.STREAM_TYPE, "3");
        speechSynthesizer.startSpeaking(text, null);
    }
```

```

    private String getResourcePath() {
        return ResourceUtil.generateResourcePath(this, ResourceUtil.RESOURCE_TYPE.assets, "tts/common.jet") +
                ";" +
                ResourceUtil.generateResourcePath(this, ResourceUtil.RESOURCE_TYPE.assets, "tts/" +
profile.code + ".jet");
    }
}
```

Message.java：

```
/**
 * 消息实体类
 */
public class Message extends RealmObject {

    public Message(){}

    public Message(String text, boolean self) {
        this.text = text;
        this.self = self;
    }

    public String text;
    public Date date = new Date();
    public boolean self;
}
```

MessageAdapter.java：

```
/**
 * 消息列表的 adapter
 */
class MessageAdapter extends RecyclerView.Adapter<MessageAdapter.MessageViewHolder> {

    private static final int ROBOT = 1;
    private static final int SELF = 2;

    private LayoutInflater inflater;

    List<Message> data = new ArrayList<>();

    private SimpleDateFormat format = new SimpleDateFormat("HH:mm", Locale.getDefault());
    MessageAdapter(Context context, List<Message> data) {
        this.inflater = LayoutInflater.from(context);
        this.data.addAll(data);
    }

    void clear() {
        data.clear();
        notifyDataSetChanged();
    }

    void addMessage(Message message) {
        data.add(message);
        notifyItemInserted(data.size());
        Realm realm = Realm.getDefaultInstance();
        realm.executeTransaction(_realm -> _realm.copyToRealm(message));
        realm.close();
    }

    @NonNull
    @Override
    public MessageViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int type) {
        int resId = type == ROBOT ? R.layout.view_message_robot_item : R.layout.view_message_self_item;
        return new MessageViewHolder(inflater.inflate(resId, viewGroup, false));
    }

    @Override
    public void onBindViewHolder(@NonNull MessageViewHolder messageViewHolder, int i) {
        Message message = data.get(i);
        TextView textView = messageViewHolder.itemView.findViewById(R.id.tv_text);
        TextView dateView = messageViewHolder.itemView.findViewById(R.id.tv_date);

        textView.setText(message.text);
        dateView.setText(format.format(message.date));
    }

    @Override
    public int getItemCount() {
        return data.size();
    }

    @Override
    public int getItemViewType(int position) {
        Message message = data.get(position);
        return message.self ? SELF : ROBOT;
    }

    class MessageViewHolder extends RecyclerView.ViewHolder {
        MessageViewHolder(@NonNull View itemView) {
            super(itemView);
        }
    }
}
```

MessagePlayProfile.java：

```
/**
 * 存放发英文配置的类
 */
public class MessagePlayProfile extends RealmObject {
    public MessagePlayProfile() {

    }

    MessagePlayProfile(boolean isLocal, String code) {
        this.isLocal = isLocal;
        this.code = code;
    }
    boolean isLocal = true;
    String code = "xiaoyan";
}
```

MyApp.java：

```
public class MyApp extends Application {
    @Override
    public void onCreate() {
        super.onCreate();


        //初始化数据库，主要是添加默认的第一条消息和默认的发音人配置
        Realm.init(this);
        Realm realm = Realm.getDefaultInstance();
        Message message = realm.where(Message.class).findFirst();
        if (message == null) {
            realm.executeTransaction(_realm ->
                    _realm.copyToRealm(new Message("你好！我是机器人，会聊天的那种，我们聊点什么吧！
", false)));
        }

        MessagePlayProfile profile = realm.where(MessagePlayProfile.class).findFirst();
        if (profile == null) {
            realm.executeTransaction(_realm ->
                    _realm.copyToRealm(new MessagePlayProfile(true,"xiaoyan")));
        }


        //发音人数据初始化
        PronunciationNames.init();


        //讯飞 SDK 初始化
        SpeechUtility.createUtility(this, SpeechConstant.APPID +"=5cff673b");
    }
}
```

NetHelper.java：

```
/**
 * 用于网络请求的辅助类
 */
public class NetHelper {

    private static NetHelper instance;

    public static NetHelper getInstance() {
        if (instance == null) {
            instance = new NetHelper();
        }
        return instance;
    }

    private Retrofit retrofit;

    private NetHelper() {
        Retrofit.Builder builder = new Retrofit.Builder();
        builder.addConverterFactory(GsonConverterFactory.create(new Gson()));
        builder.addCallAdapterFactory(RxJava2CallAdapterFactory.create());
        builder.baseUrl("http://openapi.tuling123.com/");
        retrofit = builder.build();
    }

    private final static String TU_LING_URL = "http://openapi.tuling123.com/openapi/api/v2";

    Observable<TulingResultBody> sendMessage(TulingRequstBody body) {
        return retrofit.create(TulingApi.class).sendMessage(TU_LING_URL, body)
                .subscribeOn(Schedulers.io());
    }
}
```

PronunciationNames.java：

```
/**
 * 发音人相关数据，可在这里变更替换发音人
 */
class PronunciationNames {

    static final HashMap<String, String> codeAndName = new HashMap<>();
    static final HashMap<String, String> nameAndCode = new HashMap<>();
    static void init() {
        codeAndName.put(XIAO_YAN, "小燕(普通话)");
        codeAndName.put(XIAO_FENG, "小锋(普通话)");

        codeAndName.put(XU_XIAO_BAO, "许小宝(普通话)");
        codeAndName.put(CHNEG_CHENG, "程程(普通话)");
        codeAndName.put(XIAO_RONG, "小蓉(四川话)");
        codeAndName.put(XIAO_MEI, "小梅(广东话)");
        codeAndName.put(JOHN, "John(英语)");


        nameAndCode.put("小燕(普通话)", XIAO_YAN);
        nameAndCode.put("小锋(普通话)", XIAO_FENG);

        nameAndCode.put("许小宝(普通话)", XU_XIAO_BAO);
        nameAndCode.put("程程(普通话)", CHNEG_CHENG);
        nameAndCode.put("小蓉(四川话)", XIAO_RONG);
        nameAndCode.put("小梅(广东话)", XIAO_MEI);
        nameAndCode.put("John(英语)", JOHN);
    }

    static class Local {
        static final String XIAO_YAN = "xiaoyan";
        static final String XIAO_FENG = "xiaofeng";
        static final String[] codeList = {XIAO_YAN, XIAO_FENG};
    }
```

```
    static class Remote {
        static final String XU_XIAO_BAO = "aisbabyxu";
        static final String CHNEG_CHENG = "x_chengcheng";

        static final String XIAO_RONG = "x_xiaorong";
        static final String XIAO_MEI = "x_xiaomei";
        static final String JOHN = "x_john";
        static final String[] codeList = {XU_XIAO_BAO, CHNEG_CHENG, XIAO_RONG, XIAO_MEI, JOHN};
    }
}
```

SetPronunciationActivity.java：

```
/**
 * 设置发音人的界面
 */
public class SetPronunciationActivity extends AppCompatActivity {

    private Realm realm;
    private RecyclerView recyclerView;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_set_pronunciation);

        recyclerView = findViewById(R.id.name_list);

        realm = Realm.getDefaultInstance();
        //取出保存的发音人配置
        MessagePlayProfile profile = realm.where(MessagePlayProfile.class).findFirst();
        if (profile == null) {
            return;
        }
        //根据配置显示默认显示的发音人列表
        String[] codes = profile.isLocal ? PronunciationNames.Local.codeList : PronunciationNames.Remote.codeList;
        ArrayList<String> names = new ArrayList<>();
        for (String code : codes) {
            String name = PronunciationNames.codeAndName.get(code);
            names.add(name);
        }

        NameAdapter adapter = new NameAdapter(this, names, profile.isLocal);
        recyclerView.setAdapter(adapter);

        RadioButton offline = findViewById(R.id.rb_offline);
        RadioButton online = findViewById(R.id.rb_online);


        //根据配置选中默认的发音人选项
        offline.setChecked(profile.isLocal);
        online.setChecked(!profile.isLocal);


        //离线合成点击后替换发音人列表数据为离线发音人
        offline.setOnCheckedChangeListener((buttonView, isChecked) -> {
            if (isChecked) {
                ArrayList<String> localNames = new ArrayList<>();
                for (String code : PronunciationNames.Local.codeList) {
                    String name = PronunciationNames.codeAndName.get(code);
                    localNames.add(name);
                }
                adapter.changeData(localNames, true);
            }
        });
```

```
        //在线合成点击后替换发音人列表数据为在线发音人
        online.setOnCheckedChangeListener((buttonView, isChecked) -> {
            if (isChecked) {
                ArrayList<String> remoteName = new ArrayList<>();
                for (String code : PronunciationNames.Remote.codeList) {
                    String name = PronunciationNames.codeAndName.get(code);
                    remoteName.add(name);
                }
                adapter.changeData(remoteName, false);
            }
        });
    }

    class NameAdapter extends RecyclerView.Adapter<NameAdapter.NameViewHolder> {
        private List<String> data = new ArrayList<>();
        private LayoutInflater inflater;
        private boolean isLocal;

        NameAdapter(Context context, List<String> data, boolean isLocal) {
            this.isLocal = isLocal;
            this.inflater = LayoutInflater.from(context);
            this.data.addAll(data);
        }

        void changeData(List<String> data, boolean isLocal) {
            this.isLocal = isLocal;
            this.data.clear();
            this.data.addAll(data);
            notifyDataSetChanged();
        }
```

```java
        @NonNull
        @Override
        public NameViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
            return new NameViewHolder(inflater.inflate(R.layout.view_name_item, viewGroup, false));
        }

        @Override
        public void onBindViewHolder(@NonNull NameViewHolder nameViewHolder, int i) {
            String name = data.get(i);

            MessagePlayProfile profile = realm.where(MessagePlayProfile.class).findFirst();
            if (profile == null) return;
            TextView nameView = nameViewHolder.itemView.findViewById(R.id.tv_name);
            nameView.setText(name);
            CheckBox cb = nameViewHolder.itemView.findViewById(R.id.cb);
            cb.setChecked(profile.code.equals(PronunciationNames.nameAndCode.get(name)));
            cb.setOnClickListener(v -> {
                if (cb.isChecked()) {
                    realm.executeTransaction(realm -> {
                        profile.code = PronunciationNames.nameAndCode.get(name);
                        profile.isLocal = isLocal;
                    });
                    notifyDataSetChanged();
                }
                cb.setChecked(true);
            });
        }

        @Override
        public int getItemCount() {
            return data.size();
        }

        class NameViewHolder extends RecyclerView.ViewHolder {

            NameViewHolder(@NonNull View itemView) {
                super(itemView);
            }
        }
    }
}
```

TulingApi.java：

```java
public interface TulingApi {
    @POST
    Observable<TulingResultBody> sendMessage(@Url String url, @Body TulingRequstBody postParmas);
}
```

TulingRequstBody.java：

```java
/**
 * 图灵机器人的请求实体类，部分信息已经使用默认值配置，调用时需要传递的只是 inputText
 */
class TulingRequstBody {
    private int reqType = 0;
    Perception perception = new Perception();
    private   UserInfo userInfo = new UserInfo();

    private class UserInfo {
        private   String apiKey = "02ad6413903c44eabd7663b602f052c6";
        private String userId = "460081";
    }

    class Perception {
        InputText inputText = new InputText();
        private SelfInfo selfInfo = new SelfInfo();
    }

    private class SelfInfo {
        Location location = new Location();
    }

    private class Location {

        String city = "北京";

        String province = "北京";

        String street = "天安门";

    }

    class InputText {
        String text;
    }
}
```

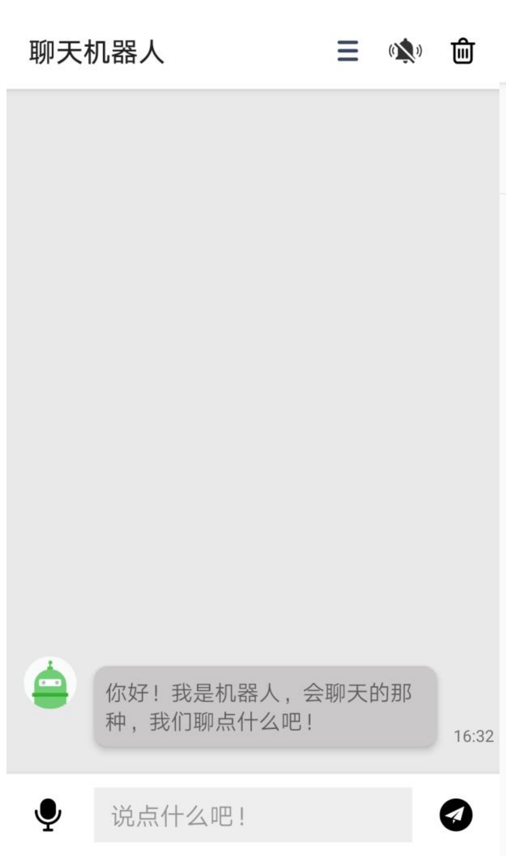
TulingResultBody.java：

```
/**
 * 图灵机器人 api 返回的实体类，主要数据在 results 的 values 中
 */
class TulingResultBody {
	Intent intent;
	List<Results> results;

	class Results {
		int groupType;
		String resultType;
		Values values;
	}

	class Values {
		String url;
		String text;
	}

	class Intent {
		int code;
		String intentName;
		String actionName;
		Parameters parameters;
	}

	class Parameters {
		String nearby_place;
	}
}
```

XFResult.java：

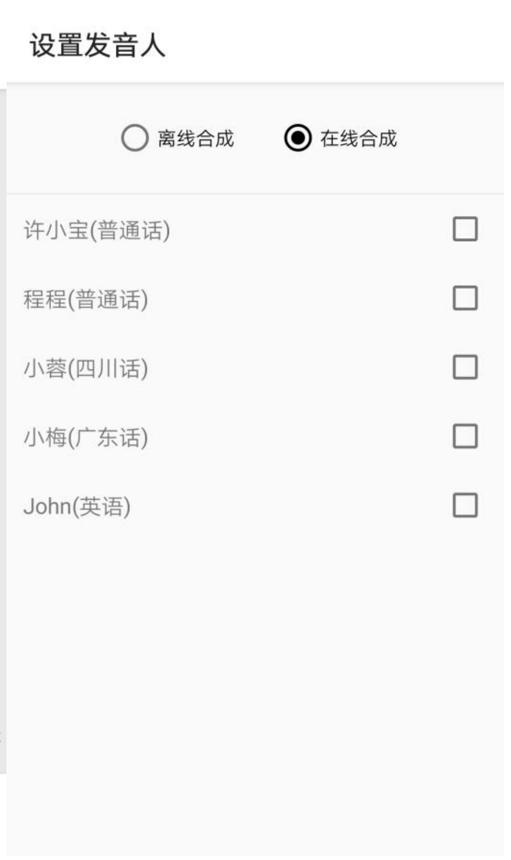
```
/**
 * 讯飞的语音合成结果实体类，包含了一些不必要参数，用 getWord 直接提取相关文本
 */
public class XFResult {
	private int sn;

	private boolean ls;

	private int bg;

	private int ed;

	List<Ws> ws;

	class Ws {
		private int bg;

		List<Cw> cw;
	}

	class Cw {
		private int sc;

		private String w;
	}

	String getWord() {
		StringBuilder builder = new StringBuilder();
		for (int i = 0; i < ws.size(); i++) {
			Ws ws = this.ws.get(i);
			for (int j = 0; j < ws.cw.size(); j++) {
				Cw cw = ws.cw.get(j);
				builder.append(cw.w);
			}
		}
		return builder.toString();
	}
}
```

三、 效果展示：


聊天机器人

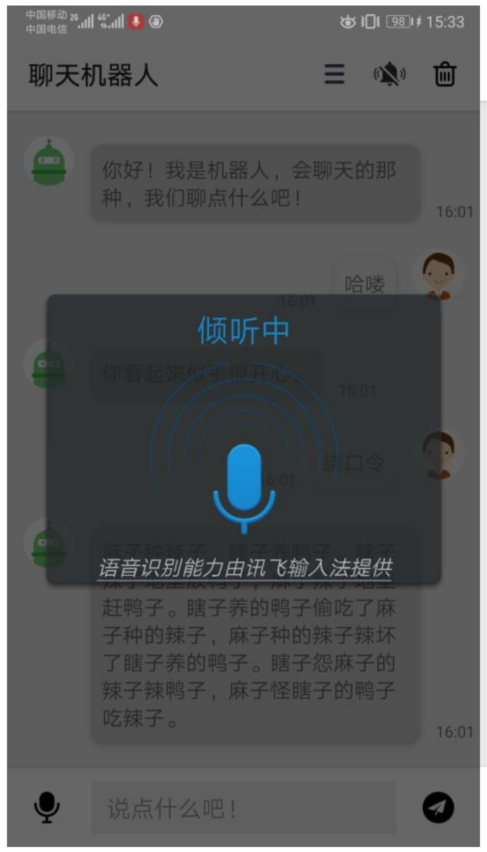
你好！我是机器人，会聊天的那种，我们聊点什么吧！ 16:32

说点什么吧！


设置发音人

○ 离线合成 [✓ 在线合成]

许小宝(普通话)	☐
程程(普通话)	☐
小蓉(四川话)	☐
小梅(广东话)	☐
John(英语)	☐


聊天机器人

你好！我是机器人，会聊天的那种，我们聊点什么吧！ 16:01

哈喽 16:01

你看起来似乎很开心 16:01

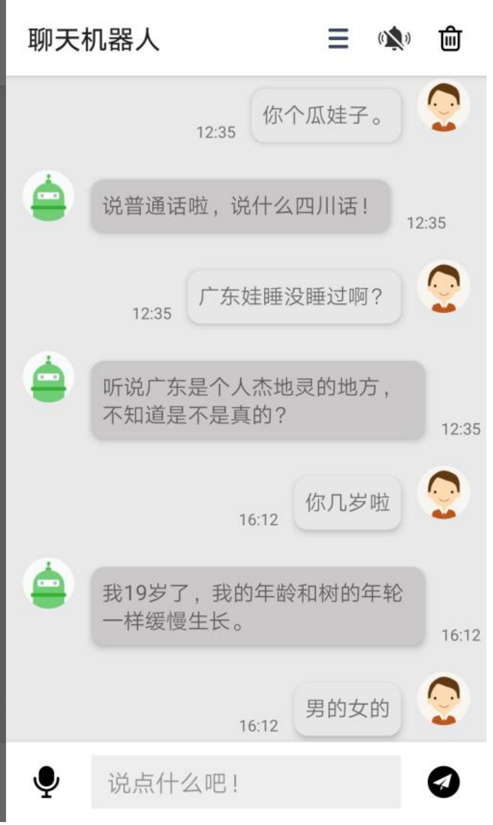
绕口令 16:01

赶鸭子。瞎子养的鸭子偷吃了麻子种的辣子，麻子种的辣子辣坏了瞎子养的鸭子。瞎子怨麻子的辣子辣鸭子，麻子怪瞎子的鸭子吃辣子。 16:01

倾听中

语音识别能力由讯飞输入法提供

说点什么吧！


聊天机器人

12:35 你个瓜娃子。

说普通话啦，说什么四川话！ 12:35

12:35 广东娃睡没睡过啊？

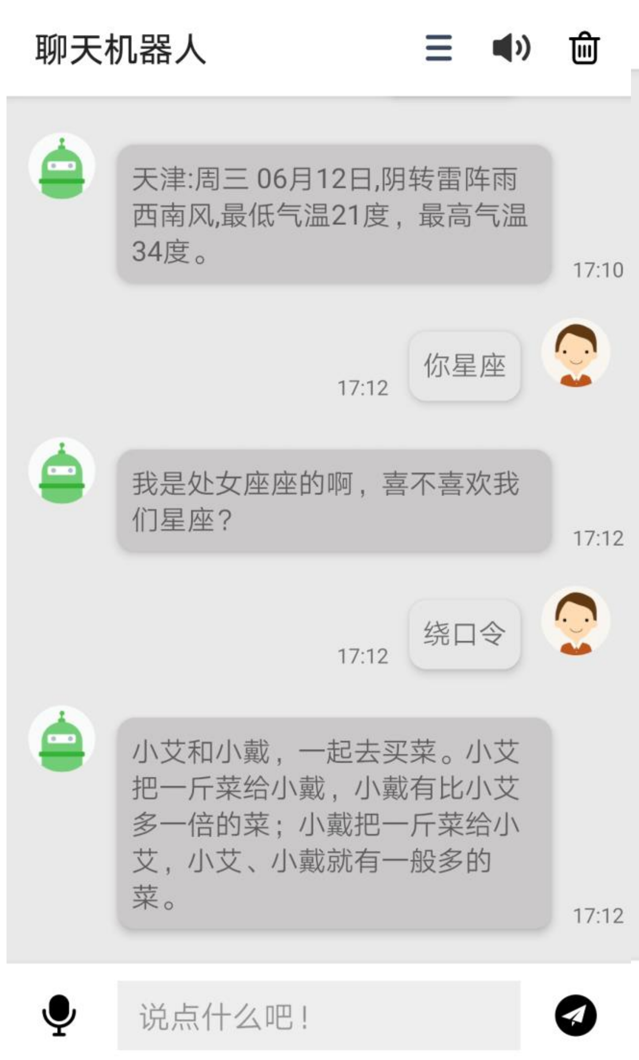
听说广东是个人杰地灵的地方，不知道是不是真的？ 12:35

16:12 你几岁啦

我19岁了，我的年龄和树的年轮一样缓慢生长。 16:12

16:12 男的女的

说点什么吧！


聊天机器人

天津:周三 06月12日,阴转雷阵雨 西南风,最低气温21度，最高气温34度。 17:10

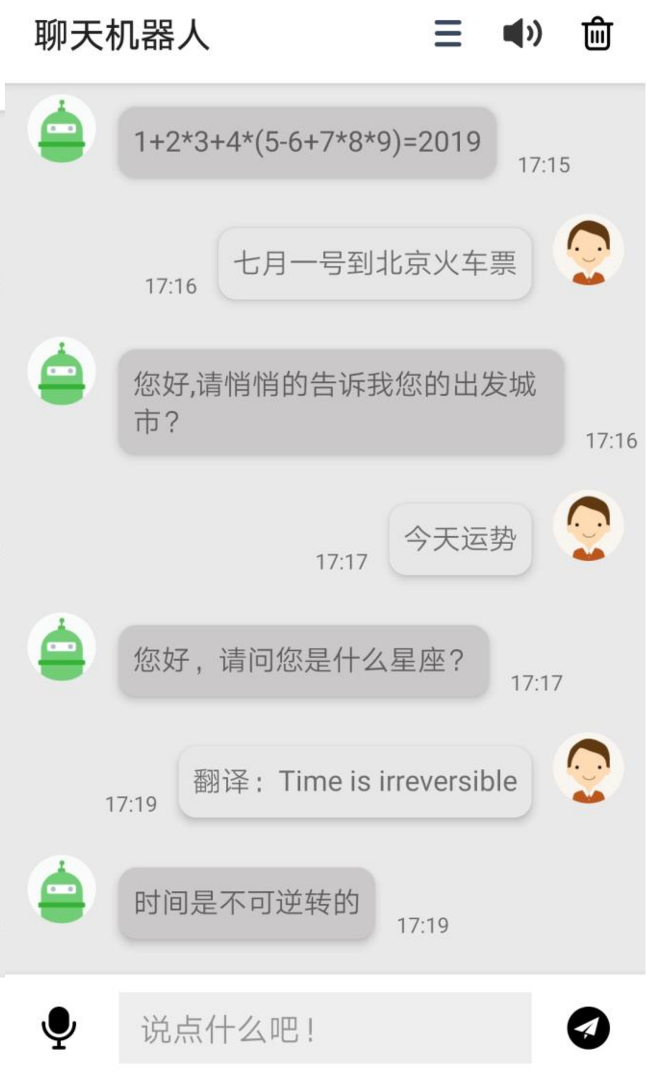
17:12 你星座

我是处女座座的啊，喜不喜欢我们星座？ 17:12

17:12 绕口令

小艾和小戴，一起去买菜。小艾把一斤菜给小戴，小戴有比小艾多一倍的菜；小戴把一斤菜给小艾，小艾、小戴就有一般多的菜。 17:12

说点什么吧！


聊天机器人

1+2*3+4*(5-6+7*8*9)=2019 17:15

17:16 七月一号到北京火车票

您好,请悄悄的告诉我您的出发城市？ 17:16

17:17 今天运势

您好，请问您是什么星座？ 17:17

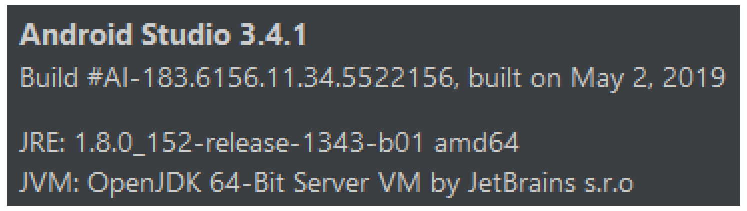
17:19 翻译：Time is irreversible

时间是不可逆转的 17:19

说点什么吧！

四、 开发环境

Windows10


```
Android Studio 3.4.1
Build #AI-183.6156.11.34.5522156, built on May 2, 2019

JRE: 1.8.0_152-release-1343-b01 amd64
JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
```

教师签字：

年　月　日