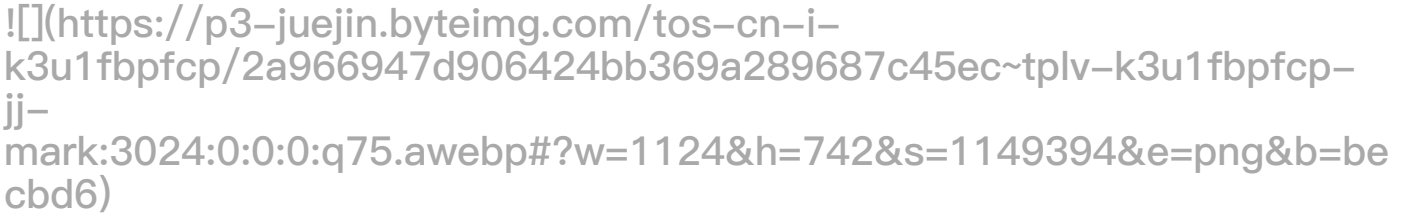
短视频配音原来如此简单

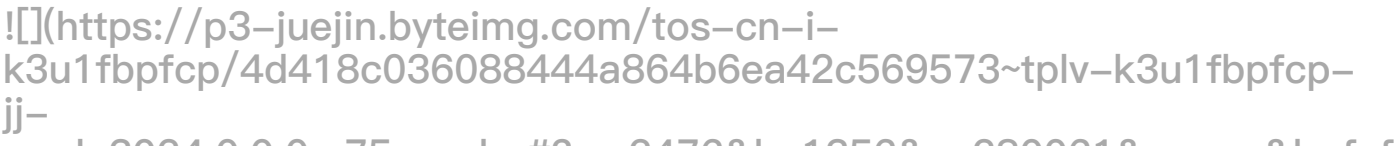
没有Ai不会写代码了

前两天淘宝购买的IDEA copilot插件的账号不能用，没有Ai的加持感觉不会写代码了。于是启用了尘封好久的通义灵码，也可能是用法不对，总感觉没有 copilot智能，毕竟廖胜于无嘛... 看着又在一行行自动生成的代码，陷入了沉思：我们是Ai的工具，还是Ai是我们工具。最近密集的面试过程中，发现大部分人也没在用，甚至都没听过这样的工具。《劝学》中有云：君子生非异也，善假于物。可能，对我这样普普通通的程序员而言，三位一体全方位拥抱，学习，改造这些工具方是良策。前一篇文章《短视频文案提取的简单实现》提到的文案提取功能，其实也是借住一些工具简单实现了，今天再来聊一聊短视频配音的简单实现。



探索配音实现

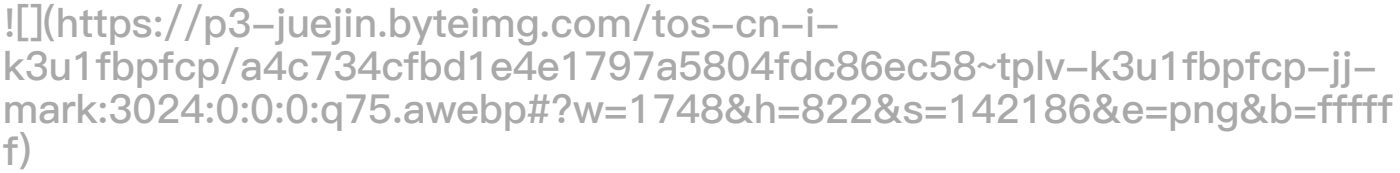
一开始看轻抖小程序上的配音功能，有停顿，有多音字，有语速等配置，顿时感觉挺有意思的。10前年做外卖配送系统时，为了方便提醒配送员抢单，用科大讯飞的TTS实现了订单语音播报，但只是简单的朗读而已。摸索了一番后，了解腾讯云已经有相关TTS接口了，看到腾讯云已经提供的能力时，我感觉基本就是调用一个接口就基本ok了， 隐约看到了我自己在小程序上实现了智能配音功能。事实证明，真是纸上得来终觉浅，绝知此事要躬行。



语音合同的核心接口比较简单，就两接口

* 基础合成（156字以内） - 同步返回
* 长语音合成（10万字以内）- 异步返回

代码实现上使用策略模式处理不同字数的场景，使用Spring Event 统一同步与异步处理逻辑，音频文件上传到cos方便下载，前端使用setTimeout 轮询查询，核心类图如下：



有了通义灵码的辅助，三下五除二，便以迅雷不及掩耳之势就写好了基础代码。这里贴下基础合成语音的代码，长语音合成就是加了一个回调url的地址，返回的是任务id, 通过回调拿到语音文件的临时地址。

```
/**
 * @Author: JJ
 * @CreateTime: 2023-11-21  09:49
 * @Description: 腾讯云tts - 一句话接口(150字以下)
 */
@Component
@Slf4j
public class SentenceTtsProcessor implements TtsProcessor {

    private static Credential cred = new
Credential(AppConstant.Tencent.asrSecretId,
AppConstant.Tencent.asrSecretKey);

    /**
     * @param complexAudioReq
```

```
     * @description: tts
     * @author: JJ
     * @date: 11/21/23 09:48
     * @param: [bytes]
     * @return: java.lang.String
     */
    @Override
    public TtsRes run(ComplexAudioReq complexAudioReq) {

        String reqId = complexAudioReq.getRequestId();
        //如果为这。生成uuid
        if (Strings.isBlank(reqId)){
            reqId = UUID.randomUUID().toString();
        }
        log.info("tts - 基础合成 {}", reqId);
        // 实例化一个http选项，可选的，没有特殊需求可以跳过
        HttpProfile httpProfile = new HttpProfile();
        httpProfile.setEndpoint("tts.tencentcloudapi.com");
        // 实例化一个client选项，可选的，没有特殊需求可以跳过
        ClientProfile clientProfile = new ClientProfile();
        clientProfile.setHttpProfile(httpProfile);
        // 实例化要请求产品的client对象,clientProfile是可选的
        TtsClient client = new TtsClient(cred, "ap-shanghai", clientProfile);
        // 实例化一个请求对象,每个接口都会对应一个request对象
        TextToVoiceRequest req = new TextToVoiceRequest();
        req.setText(complexAudioReq.getTtsText());
        req.setSessionId(reqId);
        req.setVolume(complexAudioReq.getVolume().floatValue());
        req.setSpeed(complexAudioReq.getSpeed().floatValue());
        req.setProjectId(88L);
        req.setModelType(1L);
        req.setVoiceType(complexAudioReq.getVoiceTypeId());
        req.setPrimaryLanguage(1L);
        req.setEnableSubtitle(false);
req.setEmotionCategory(EmotionMap.getEmotion(complexAudioReq.getE
motionCategory()));
        req.setEmotionIntensity(complexAudioReq.getEmotionIntensity());

        try {
            // 返回的resp是一个TextToVoiceResponse的实例，与请求对象对
应
            TextToVoiceResponse resp = client.TextToVoice(req);
            log.info("tts - 基础合成完成 SessionId={}，req={}", reqId,
resp.getRequestId());
            TtsRes ttsRes = TtsRes.builder()
                    .ttsType(TtsTypeEnum.SENTENCE.code())
                    .requestId(resp.getRequestId())
                    .data(resp.getAudio())
```

```
                .build();

            return ttsRes;
        } catch (TencentCloudSDKException e) {
            log.error("一句话tts失败:{}",e);
            throw new BusinessException(SENTENCE_TTS_ERROR.code(),
SENTENCE_TTS_ERROR.desc());
        }
    }

    /**
     * @param req
     * @description: filter 根据参数选
     * @author: JJ
     * @date: 3/3/24 18:54
     * @param:
     * @return:
     */
    @Override
    public Boolean filter(ComplexAudioReq req) {
        // 字数小于150
        if (req.getTtsTextLength() <
AppConstant.Tencent.Sentence_TTS_Max_Word_Count){
            return true;
        }
        return false;
    }
}
```

收到合成成功的回调后，发送事件，监听器异步处理。

```
            //发送异步事件，上传cos
            AudioUploadCosEvent uploadCosEvent =
AudioUploadCosEvent.builder()
                    .eventTime(System.currentTimeMillis() / 1000)
                    .recordId(usageRecordEntity.getId())
                    .remote(true)
                    .dataUrl(req.getResultUrl()).build();

            applicationContext.publishEvent(uploadCosEvent);
```

```

事件监听核心逻辑就是上传cos，并修改合成记录状态，代码非常简单，大致如下：

```
 /**
     * 音频异步上传cos
     * @param event
     */
    @Async
    @EventListener
    public void audioUploadCos(AudioUploadCosEvent event) {

            // 上传cos
            InputStream inputStream = null;
            //根据是否远程走不同的逻辑
            if (event.isRemote()){
               //跟url 下载 生成inputStream
                log.info("开始下载音频{} ", updateModel.getId());
                MediaDownloadReq videoReq = new MediaDownloadReq();
                videoReq.setUrl(event.getDataUrl());
                videoReq.setTargetFileSuffix("wav");
                inputStream = mediaDownloader.run(videoReq);

            }else {
                byte[] decodedBytes = Base64.decode(event.getData());
                inputStream = new ByteArrayInputStream(decodedBytes);
            }

            //上传音频到cos
            String yyyyMM = DateUtils.dateFormatDateTime(new Date(),
DateUtils.formatyyyyMM);
            String path =
"/lp/audio/"+yyyyMM+"/"+updateModel.getId()+".wav";
            OssUploadResponse ossUploadResponse =
OSSFactory.build().upload(inputStream, path);
            // 关闭InputStream
            inputStream.close();
            log.info("上传音频到cos完成{}", ossUploadResponse.getUrl());
            // 修改记录状态
```
```

```
    }
```

原来还在半山腰
=======

最近在搞2024团队规划，Boss希望我们能预估到大致人日，之前也有过这样的预估实际上每次都预估偏差都不小，毕竟人日可能都在细节上。最后退而求其次，预估下两个月的人日，没有PRD，没有技术方案大概率预估的人日可能只会在半山腰，就如同配音的功能写到这里，我以为已经“会当临绝顶，一览众山小了”，哪知道一山还有一山高。

第一难就是多音字，要得到正确的发音，就需要明确指出发音与声调。比如对于腾讯云的语音合成接口支持 SSML 标记语言，比如我们需要让“长”发音为“zhang”，就需要做这样的标记。

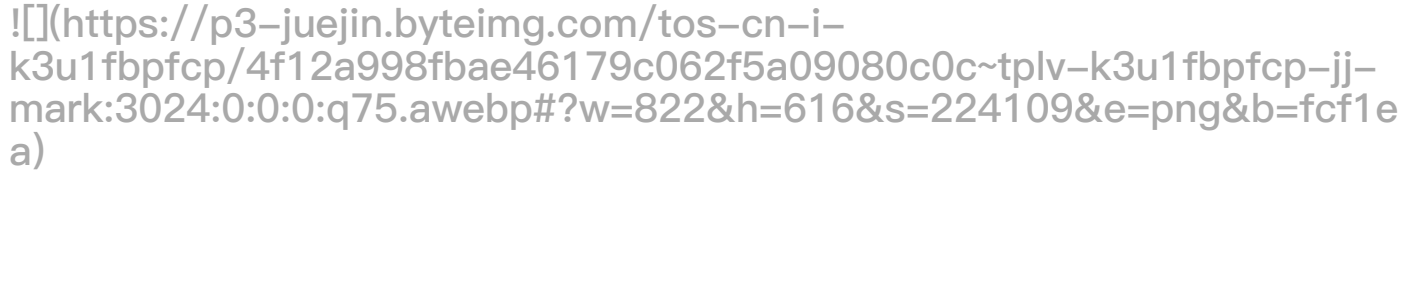
```
<speak><phoneme alphabet=”py” ph=”zhang3”>长</phoneme></speak>
```

对于后端而言，这个是简单的，通过pinyin4j可以快速找出一段文本中的多音字及其所有读音，几行代码就解决了。

```
 /**
     * 多音字检测
     * @param req
```

```
     * @return
     * @throws Exception
     */
    public List<PolyphoneVo> run(PolyphoneQuery req) {
        //遍历字符串，找出多音字
        char[] chars = req.getTtsText().toCharArray();
        List<PolyphoneVo> polyphoneVoList = new ArrayList<>();
        Set<String> polyphoneWordSet = new HashSet<>();
        for (char c : chars) {
            if((c >= 0x4e00)&&(c <= 0x9fbb)) {
                String[] pinyinList = PinyinHelper.toHanyuPinyinStringArray(c);
                if (pinyinList != null && pinyinList.length > 1 &&
!polyphoneWordSet.contains(c+"")) {
                    PolyphoneVo polyphoneVo = new PolyphoneVo();
                    polyphoneVo.setWord(c+"");
                    polyphoneVo.setReadList(Arrays.asList(pinyinList));
                    polyphoneVoList.add(polyphoneVo);
                    polyphoneWordSet.add(c+"");
                }
            }
        }
        return polyphoneVoList;
    }
```

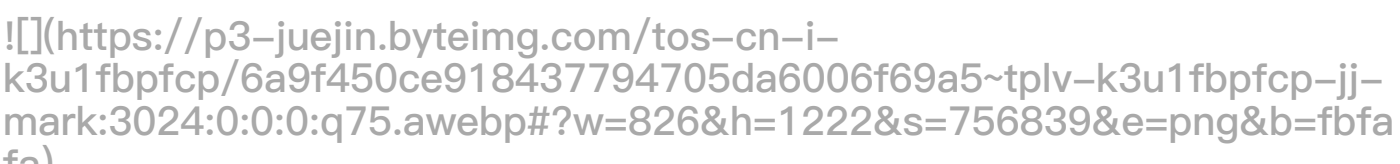
```

```

```

前端难点在于如何让多音字可以点击以及点击后的交互，以及SSML 标记语言替换。给大家来两张效果图，有兴趣的可以脑补下前端实现。主要两个地方注意下：显示文本的数据结构和正则表达式。


```

行文至此，停顿有了，发音正常了，音频文件也有了，播放音频也正常了，只是进度条着实费了一些神，官方没有进度条的实现，最后用 van-slider 模拟实现了。唯一bug是无法获取正确的音时长，正当一筹莫展时，又有一个问题出现了：mp3格式无法正常在小程序中保存文件，官方文档中的描述确实只支持mp4文件。一番斗争后，觉得这个问题更为重要，于是又开始了摸索。

音频转视频的意外收获
==========

mp3转成mp4，还是javaCV，有了之前的视频中分离音频的经历，这次就顺利多了。唯一的问题就是：视频文件的祯没有内容，所以是一片黑。于是想着用一张固定的图片做为祯内容。中间最麻烦的是音频与祯如何同步。最后还是与GPT4进行了一次多轮对话才完美了解决了。 用小程序二维码图做为默认的祯，效果如下图。在转换过程中，又根据 FFmpegFrameGrabber.getLengthInTime 获取到了音频进长，顺道解决了前面无法解决的问题。真可谓是“无心插柳柳成荫”。代码与上一篇文章中的文案提取的代码基本雷同，就不贴了。



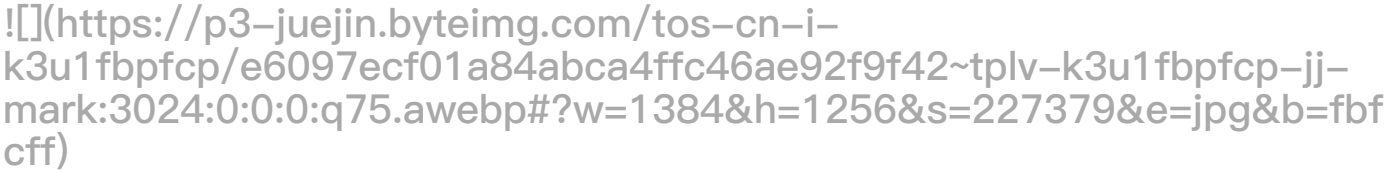
写在最后
====

写到这里，坑算是基本都趟过了。虽说实际所花的时间早已远超之前的计划了，好在出来效果还不错，也顺道补充了一些基本见识，也是不错了。再回来Ai辅助编程的话题，Ai知道的东西很多，会越来越多，如果提升个人思维能力，如何利用AI 估计很快会成为大部分的程序员了必修课了。

有兴趣的同学可以扫码体验下小程序。

小程序名称 ： 智能配音实用工具；

小程序二维码：



原文链接: https://juejin.cn/post/7354309874669813798