Aplicação Java para Transcrição de Áudio com Vosk e FFmpeg no Windows

Transcrever áudios em diversos formatos e tamanhos para texto é uma tarefa essencial em áreas como análise de dados, legendagem, criação de audiolivros e acessibilidade digital.

Este guia detalha como criar uma aplicação Java para realizar essa tarefa de forma eficiente, utilizando as bibliotecas **Vosk** (reconhecimento de fala offline) e **FFmpeg** (conversão e manipulação de áudio).

A solução suporta arquivos de áudio longos ao dividi-los em blocos menores, processando-os de maneira incremental para evitar sobrecarga de memória e CPU.

Objetivos

1. **Suportar diversos formatos de áudio**: MP3, MP4, WAV, AAC, OGG, e outros.
2. **Lidar com áudios longos**: Dividir arquivos grandes em blocos menores e transcrevê-los sequencialmente.
3. **Fornecer uma saída incremental**: Salvar a transcrição de cada bloco em um arquivo de texto para facilitar o manuseio.
4. **Ser multiplataforma e eficiente**: Usar Java, uma linguagem amplamente compatível e versátil.

Requisitos do Sistema

Ferramentas Necessárias

1. **Java Development Kit (JDK)**:
   - Baixe e instale o **JDK 11 ou superior**.
   - Link para download.
   - Após a instalação, configure a variável de ambiente JAVA_HOME e adicione o diretório bin ao PATH.

Verificação: java -version

Saída esperada: java version "11.0.x"

2. **Spring Tool Suite (STS)**:
   - Baixe e instale o STS:
     - Link para download.
   - Recomendação: **STS 4**, compatível com Maven e JDK 11+.
3. **Maven**:
   - Instalado automaticamente com o STS.
4. **FFmpeg**:
   - Baixe a versão estática do FFmpeg para Windows:
     - Link para download.
   - Extraia o arquivo e adicione o diretório bin ao PATH.

Verificação

ffmpeg -version

Saída esperada:

ffmpeg version x.x.x

5. **Modelo Vosk para Português**:
   - Baixe o modelo de português em: Vosk Models.
   - Extraia o conteúdo para a pasta src/main/resources/model-pt no projeto.

Passo a Passo

1. Criando o Projeto no STS

1. Abra o **Spring Tool Suite** e clique em File > New > Maven Project.
2. Na janela que abrir:
   - Marque Create a simple project (skip archetype selection).
   - Clique em **Next**.
3. Preencha os campos:
   - **Group ID**: com.transcription
   - **Artifact ID**: audio-transcriber
   - **Version**: 1.0
   - **Package**: com.transcription
4. Clique em **Finish**.

2. Configurando o Arquivo pom.xml

Abra o arquivo pom.xml e adicione as dependências abaixo:

```
xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.transcription</groupId>
   <artifactId>audio-transcriber</artifactId>
   <version>1.0</version>

   <dependencies>
      <!-- Vosk Speech-to-Text -->
      <dependency>
         <groupId>com.alphacephei</groupId>
         <artifactId>vosk</artifactId>
         <version>1.0.10</version>
      </dependency>

      <!-- FFmpeg Executor -->
      <dependency>
         <groupId>net.bramp.ffmpeg</groupId>
         <artifactId>ffmpeg</artifactId>
         <version>0.6.2</version>
      </dependency>
   </dependencies>
</project>
```

Após salvar, o Maven baixará automaticamente as dependências.

3. Criando a Classe Principal

1. No diretório src/main/java, navegue até o pacote com.transcription.
2. Crie uma classe chamada AudioTranscriber.

4. Código Completo

Cole o código abaixo na classe AudioTranscriber:

```
import org.vosk.Model;
import org.vosk.Recognizer;
import net.bramp.ffmpeg.FFmpeg;
import net.bramp.ffmpeg.FFmpegExecutor;

import java.io.*;

public class AudioTranscriber {
```

```
    public static void main(String[] args) {
        try {
            // Caminho do modelo Vosk para Português
            String modelPath = "src/main/resources/model-pt";

            // Caminho do arquivo de entrada
            String inputAudioPath = "audiolivro.mp3";

            // Caminho base para os blocos de áudio (WAV)
            String outputWavBasePath = "audiolivro_part";

            // Caminho do arquivo de saída da transcrição
            String transcriptionFilePath = "transcricao.txt";

            // Dividir o áudio em blocos de 10 minutos
            divideAudioIntoBlocks(inputAudioPath, outputWavBasePath, 600);

            // Transcrever cada bloco de áudio
            transcribeAudioBlocks(outputWavBasePath, modelPath, transcriptionFilePath);

            System.out.println("Transcrição salva em: " + transcriptionFilePath);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void divideAudioIntoBlocks(String inputPath, String outputBasePath, int
segmentDuration) throws Exception {
        String ffmpegPath = "ffmpeg";

        FFmpeg ffmpeg = new FFmpeg(ffmpegPath);
        FFmpegExecutor executor = new FFmpegExecutor(ffmpeg);

        executor.createJob(
            ffmpeg.format("wav")
                .addInput(new File(inputPath).getAbsolutePath())
                .addOutput(outputBasePath + "%03d.wav")
                .setDuration(segmentDuration)
                .setAudioChannels(1)
                .setAudioSampleRate(16000)
        ).run();

        System.out.println("Arquivo dividido em blocos de " + segmentDuration + " segundos.");
    }

    private static void transcribeAudioBlocks(String outputBasePath, String modelPath, String
transcriptionPath) throws Exception {
        Model model = new Model(modelPath);

        try (FileWriter writer = new FileWriter(transcriptionPath, true)) {
            int blockIndex = 0;

            while (true) {
                String blockPath = String.format(outputBasePath + "%03d.wav", blockIndex);
                File blockFile = new File(blockPath);

                if (!blockFile.exists()) break;

                try (FileInputStream audioStream = new FileInputStream(blockFile)) {
```

```
                Recognizer recognizer = new Recognizer(model, 16000);
                byte[] buffer = new byte[4096];
                int bytesRead;

                while ((bytesRead = audioStream.read(buffer)) != -1) {
                    if (recognizer.acceptWaveForm(buffer, bytesRead)) {
                        String partialResult = recognizer.getResult();
                        writer.write(partialResult + System.lineSeparator());
                    }
                }

                String finalResult = recognizer.getFinalResult();
                writer.write(finalResult + System.lineSeparator());
            }

            System.out.println("Bloco " + blockIndex + " transcrito.");
            blockIndex++;
        }
    }

    System.out.println("Todos os blocos foram transcritos.");
  }
}
```

5. Executando o Projeto

1. Coloque o arquivo de áudio (ex.: audiolivro.mp3) na pasta raiz do projeto.
2. Clique com o botão direito na classe AudioTranscriber e selecione Run As > Java Application.
3. Após a execução:
    - Arquivos WAV segmentados (ex.: audiolivro_part001.wav) serão criados.
    - A transcrição completa será salva no arquivo transcricao.txt.

Entrada e Saída

Entrada (Arquivo de Áudio)

- **Formatos Suportados**: MP3, MP4, WAV, AAC, OGG, etc.
- **Tamanho**: Sem limite (dividido em blocos automaticamente).

Saída (Arquivo de Texto)

- Local: transcricao.txt.
- Conteúdo: Transcrição do áudio em formato JSON.

Este guia apresenta uma solução robusta e eficiente para transcrição de áudio com Java, adaptável a diferentes formatos e tamanhos de arquivo.

A divisão em blocos garante que áudios longos sejam processados sem sobrecarregar a memória, enquanto a saída incremental facilita o manuseio dos resultados.

Vantagens:

- **Escalabilidade**: Suporte a áudios longos, divididos em partes menores.
- **Versatilidade**: Suporte a múltiplos formatos de áudio por meio do FFmpeg.
- **Offline**: O uso do Vosk permite transcrição sem necessidade de conexão com a internet.

Possíveis Melhorias:

- Adicionar suporte a paralelismo para processar múltiplos blocos simultaneamente.
- Incluir uma interface gráfica para facilitar o uso.

Com esta base, você pode expandir a solução para atender a cenários específicos, como legendagem automática, criação de audiolivros e análises avançadas.

EducaCiência FastCode para a comunidade