Voice Interaction Implementation for CrewAI Research Agents

Implementation Overview

AI notice

This implementation was created by using AI IDE tool Cursor. Given human's implementation prompt, Cursor created the implementation plan, than followed it to add TTS and STT capabilities - both architecture and implementation decisions. Cursor also was used in debugging the code and fixing implementation issues (like not being able to record audio within Jupyter implementation within Cursor) as well as in drafting this implementation summary.

Libraries and APIs Used

Core Libraries:

- `openai>=1.0.0` - OpenAI Python SDK for Whisper and TTS APIs
- `ipywidgets>=8.0.0` - Jupyter notebook widgets (initially attempted for recording inline within Jupyter, currently not working properly)
- `pydub>=0.25.1` - Audio processing utilities
- `IPython.display` - Audio playback in notebooks

APIs and SDKs:

- **OpenAI Whisper API** (`whisper-1` model) - Speech-to-text transcription
- **OpenAI TTS API** (`tts-1` model, `alloy` voice) - Text-to-speech synthesis
- **OpenAI Python SDK** - Client library for API interactions

Technical Implementation

Voice Input (Speech-to-Text):

```
def transcribe_audio(audio_data, language="en"):
    client = OpenAI(api_key=get_openai_api_key())
    transcript = client.audio.transcriptions.create(
        model="whisper-1",
        file=audio_file,
        language=language,  # Force English to prevent unwanted translations
        response_format="text"
    )
    return transcript.strip()
```

Voice Output (Text-to-Speech):

```
def text_to_speech(text, voice="alloy"):
    client = OpenAI(api_key=get_openai_api_key())
    response = client.audio.speech.create(
        model="tts-1",
        voice=voice,
        input=text
    )
    return response.content
```

Audio Playback and Storage:

```
def play_and_save_audio(audio_content, filename_prefix="audio_output"):
    # Auto-play in notebook
    display(Audio(audio_content, autoplay=True))
    # Save timestamped file
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    filename = f"{filename_prefix}_{timestamp}.mp3"
    # Write to file system
```

Example Run: Fintech Research

Input Process

1. **Industry Input**: Voice recording saying "fintech" → Transcribed as "fintech"
2. **Region Input**: Voice recording saying "United States" → Transcribed as "United States"

3. **Time Window Input**: Voice recording saying "last 12 months" → Transcribed as "last 12 months"

Research Execution

- CrewAI agents processed the inputs through 3 stages:
  - Industry taxonomist mapped fintech sub-industries
  - Research analyst found companies in each sub-industry
  - Classifier/ranker produced final ranked list

Output Results

Voice Output: In-line audio playback of spoken results + a separate audio file:

> "Research completed for fintech in United States over last 12 months. Here are the top companies found: 1. PayPal Holdings, Inc., Payments & Payment Processing, Big Tech. 2. Block, Inc., Payments & Payment Processing, Big Tech. 3. Global Payments Inc., Payments & Payment Processing, Big Tech..."

Saved File example in GitHub repo: `research_results_20241220_143022.mp3`

Key Insights and Lessons Learned

Technical Challenges Overcome

1. **Real-time Recording Limitations**
   - **Issue**: `ipywidgets.AudioRecorder` doesn't exist in current versions of ipywidgets, a library suggested by Cursor
   - **Solution**: Switched to file upload approach - users should record audio externally, upload files separately and provide a link to the recording
   - **Insight**: Jupyter notebook environments have limited real-time audio recording capabilities
2. **Unwanted Language Translation**
   - **Issue**: Whisper API was auto-detecting language and transcribing / translating to Russian without any specific prompt
   - **Solution**: Added explicit `language="en"` parameter to force English transcription
   - **Insight**: Always specify language parameter to prevent unexpected translations
3. **Rich Console Conflicts**
   - **Issue**: CrewAI's Rich console library caused recursion errors with `print()` statements
   - **Solution**: Replaced all `print()` with `sys.stdout.write()` and `sys.stdout.flush()`
   - **Insight**: Rich console libraries can conflict with standard Python output in complex environments
4. **Input Hanging Issues**
   - **Issue**: Notebook cells would hang waiting for user input
   - **Solution**: Added comprehensive error handling with `try/except` blocks and `KeyboardInterrupt` handling
   - **Insight**: Robust error handling is essential for interactive notebook experiences

User Experience Improvements

- **Simplified Workflow**: Removed language selection dialogue, defaulting to English
- **Graceful Fallbacks**: If voice input fails, automatically falls back to text input
- **Clear Feedback**: Users receive immediate confirmation of transcription results
- **File Management**: Audio files are automatically saved with timestamps for later reference

Performance Observations

- **Transcription Accuracy**: Whisper API performed excellently with clear English speech
- **Processing Speed**: Voice input added ~2-3 seconds per prompt (file upload + transcription)
- **Audio Quality**: TTS output was clear and natural-sounding
- **File Sizes**: Typical audio files were 50-200KB, TTS output ~100-500KB
- **Overall user experience**: Current implementation may feel tedious, since it requires a lot of user inputs: first, user needs to specify is they're using voice or text inputs in general, second, they need to provide filepath to the recording, third - they need to do this three times in a row for three inputs.

Future Enhancements

- **Batch Processing**: Allow multiple voice inputs in sequence without re-prompting
- **Direct Recording**: Investigate browser-based audio recording APIs for real-time capture
- **Language Support**: Add back language selection for non-English users
- **Audio Compression**: Optimize file sizes for better performance