From Podcast to Podcast: Automated Content Localization Using OpenAI API Stack

How to Automate Podcast Translations with Whisper, GPT-4o, and TTS-1

Why Automate Podcast Localization?

Make content accessible to a wider audience.

Automating podcast localization makes content accessible to global audiences, breaking language barriers and increasing engagement.

Maintain tone and style in translation.

Translation isn't just about words; it's about preserving the original tone, humor, and style, which is done using GPT-4o.

Reduce manual effort while ensuring high-quality results.

Automation minimizes human intervention, saving time and costs, while still ensuring high-quality results.

The Automated Podcast Localization Pipeline

1. **Podcast Downloader:** Fetches metadata and audio.
2. **Transcription:** Converts speech to text using Whisper.
3. **Text Processing:** Enhances text and translates it using GPT-4o.
4. **Speech Synthesis:** Converts translated text into Russian audio using TTS-1.
5. **Audio Assembly:** Merges translated audio files.
6. **RSS Generation:** Creates an RSS feed for the translated podcast.

System Architecture

Linear pipeline:

Key Technologies

Kotlin:

Core language, chosen for clarity and compatibility.

Podcast4j:

Simplifies fetching podcasts from podcastindex.org.

OpenAI API:

- Whisper-1: Transcribes audio accurately.
- GPT-4o: Enhances and translates text.
- TTS-1: Converts text to speech.

OkHttp (via Ktor):

Manages HTTP requests.

Jackson:

Handles JSON data.

XML APIs:

Builds the RSS feed.

FFmpeg (planned):

Will enhance audio merging in the future.

Fetching Podcast Metadata and Audio

Use Podcast4j to fetch episode data from PodcastIndex.org.

Podcast4j queries PodcastIndex.org to retrieve essential episode information, including title, description, publish date, and audio file location.

Download MP3 files for processing.

After retrieving the metadata, Podcast4j downloads the corresponding MP3 files, providing the core content for processing in the subsequent stages of the workflow.

Transcription Process

Whisper-1 API converts audio to English text.

Works well for most recordings under 25 MB.

Longer episodes may require splitting.

Improving and Translating Transcription

First, GPT-4o enhances grammar, punctuation, and readability.

```
val prompt = """
   Context: this is a transcription of a podcast in $sourceLanguage, fix the grammar and punctuation,
   or translate it to $sourceLanguage if it's in a wrong language.
   Detect where podcast starts and cut unrelated content at start.

   Output format: {{formatted text In $sourceLanguage}}. Output with no introduction, only output text.

   Input: $transcription
""".trimIndent()
```

Then translates text to Russian while maintaining tone.

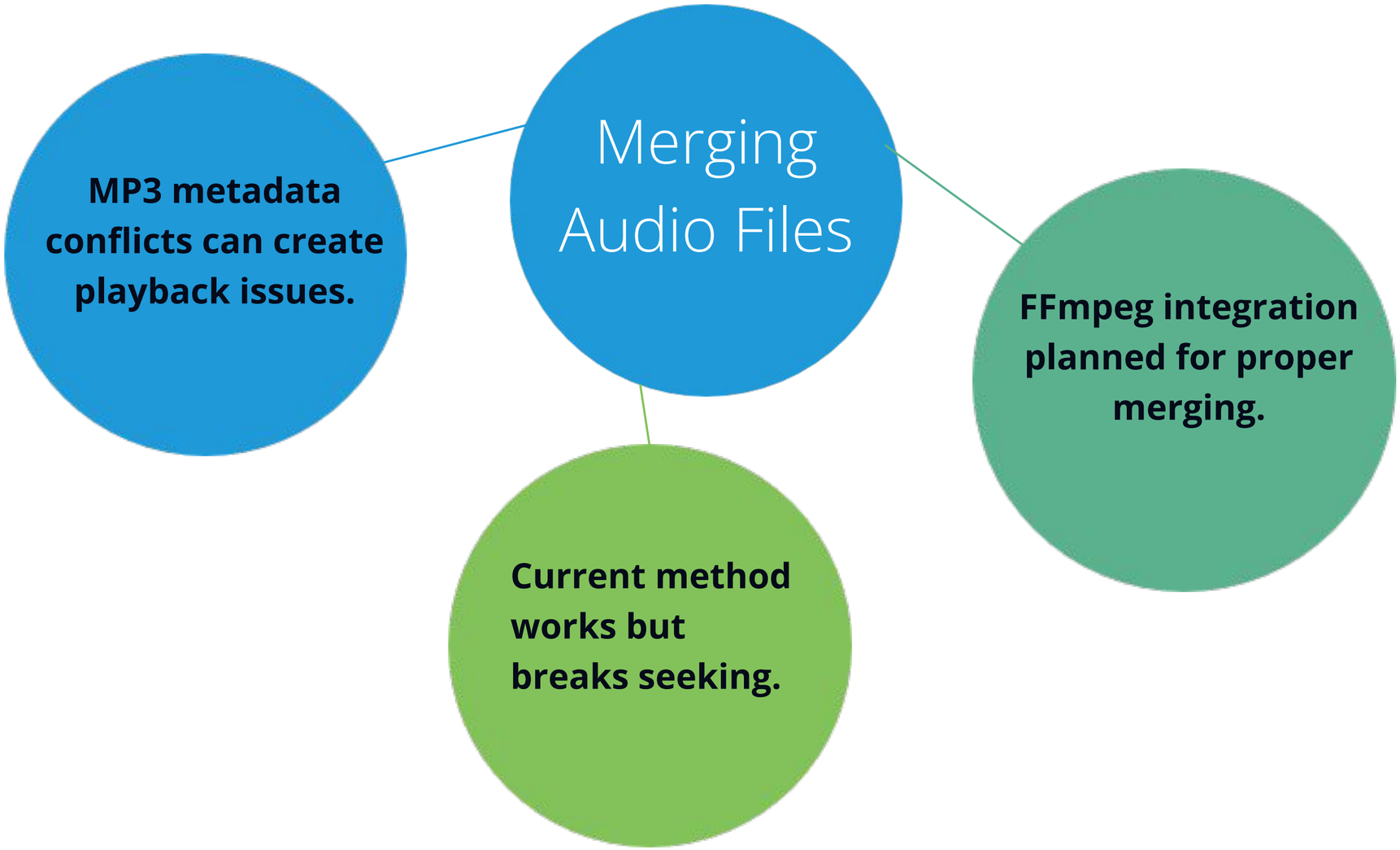
```
val prompt = """
   Translate text below to the $targetLanguage language.
   Keep the translation as close to the original in tone and style as you can.

   Text: $textToTranslate
""".trimIndent()
```

Converting Text to Speech

Russian synthesis	Challenges	Example
Russian audio generated using OpenAI's TTS-1.	Limited voice options, slight American accent.	A lively Nature Podcast host gets an upbeat TTS-1 voice saying "Здравствуйте" with a slight American twang.

Audio Assembly Challenges



Publishing the Translated Podcast

Two important aspects of the process for distributing the translated podcast:

XML-based RSS feed creation.

Translated metadata for better discoverability.

Example:

Nature Podcast RSS title changes from “From viral variants…” to “От вирусных вариантов…”.

Overcoming Technical Hurdles

File size limits in Whisper.

Style preservation in translation.

Audio merging complexities.

Optimizing GPT-4o for Translation

Grammar correction prompt.

Handling mixed-language content.

Tone preservation techniques.

Example

- **Input:** “AI is cool, and in Russian, круто!”
- **Transcript:** “AI is cool, and in Russian, kruto!”
- **Translated:** “ИИ крутой, и по-русски тоже круто!”

Maintaining Speaker's Tone

Example of TTS-1 Voice Selection Limitations:

The Russian TTS-1 voices have an American accent, making them sound slightly non-native. While usable, there's room for improvement, such as making a lively host sound less twangy when saying "Здравствуйте."

01

Handling Podcast-Specific Content:

All parts of the podcast, including intros, outros, and ads, are translated. In the future, we plan to localize ads for Russian audiences to make them more relevant.

02

Before/After Comparison Example:

Original: "AI is transforming science..."

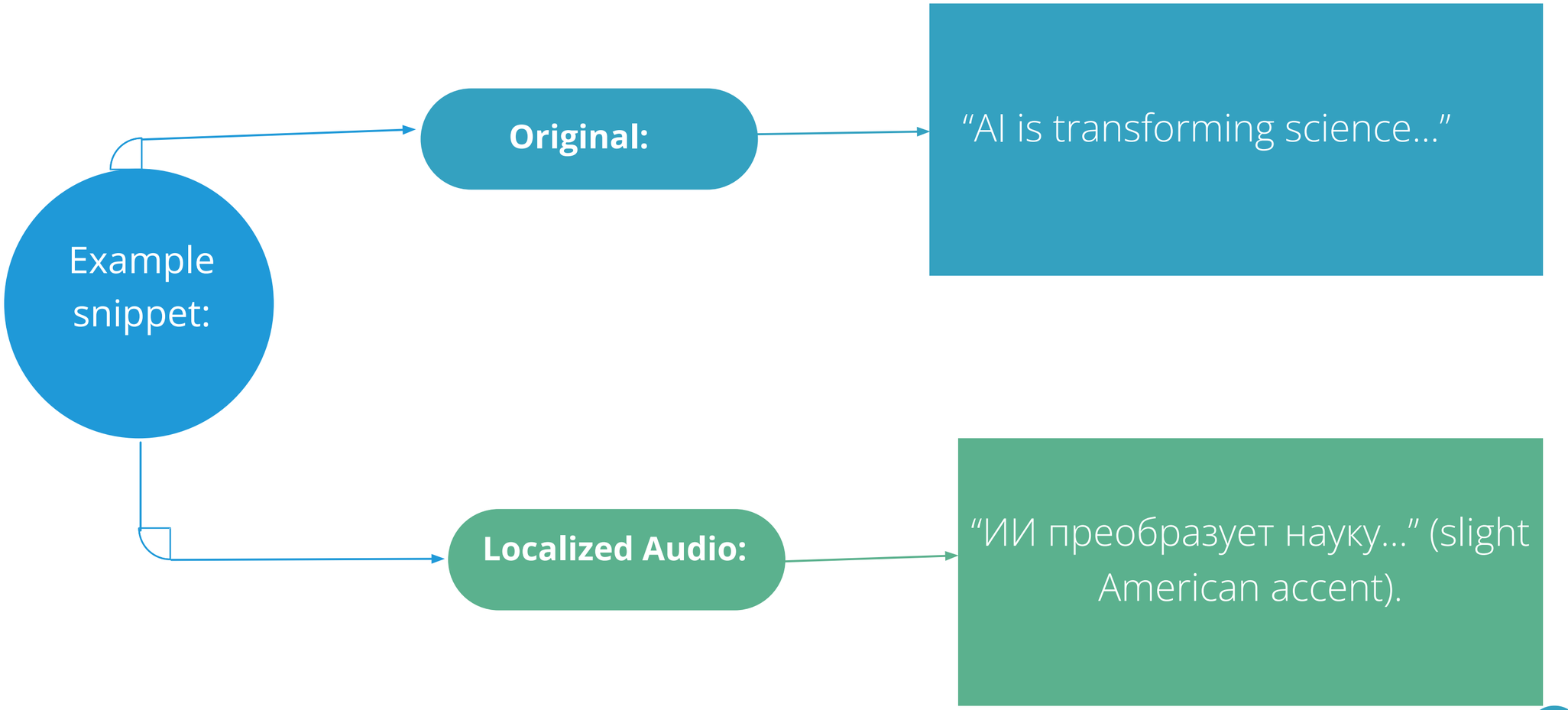
Localized Audio: "*ИИ* преобразует науку..." (with a slight American accent).

03

Ensuring Translation Accuracy

BLEU scores provide a quantitative measure of translation accuracy, while native speaker feedback ensures that the translation sounds natural and culturally appropriate. These improvements will speed up the review process and help maintain high translation quality as the system scales.

Before & After Comparison



Pre-recorded Sample Transformation

Show a Nature Podcast snippet:

- Original: “From viral variants to devastating storms…”
- Localized: “От вирусных вариантов до разрушительных штормов…”

Side-by-Side Transcript Comparison

Side-by-Side Transcript Comparison

- Original: “AI is changing the world”
- Fixed: “AI is changing the world.”
- Translated: “ИИ меняет мир.”

Key Code Snippets Showcase (Downloading the Podcast)

```
fun downloadPodcastEpisodes(podcastId: Int): List<Pair<Episode, Path>> {
    val podcast = client.podcastService.getPodcastByFeedId(podcastId)
    val episodes = client.episodeService
        .getEpisodesByFeedId(ByFeedIdArg.builder().id(podcast.id).build())

    return episodes.mapNotNull { e ->
        val mp3Path = tryDownloadEpisode(podcast, e)
        mp3Path?.let { e to mp3Path }
    }
}
```

Transcribing Audio with Whisper

```
suspend fun transcribeAudio(audioFilePath: Path): String {
    val audioFile = FileSource(KxPath(audioFilePath.toFile().toString()))

    val request = TranscriptionRequest(
        audio = audioFile,
        model = ModelId("whisper-1")
    )

    return openAIClient.transcription(request).text
}
```

Improving and Translating Text with GPT-4o

```
suspend fun improveTranscription(transcription: String, sourceLanguage: String): String {
    val prompt = """
        Context: this is a transcription of a podcast in $sourceLanguage, fix the grammar and punctuation,
        or translate it to $sourceLanguage if it's in a wrong language.
        Detect where podcast starts and cut unrelated content at start.

        Output format: {{formatted text In $sourceLanguage}}. Output with no introduction, only output text.

        Input: $transcription
    """.trimIndent()

    val request = ChatCompletionRequest(
        model = ModelId("gpt-4o"),
        messages = listOf(
            ChatMessage(
                role = ChatRole.System,
                content = prompt
            )
        ),
    )

    val response = openAIClient.chatCompletion(request)

    return response.choices.first().message.content!!
}
```

Generating Speech with TTS-1

```
suspend fun textToSpeech(translatedText: String, fileName: String): ByteArray {
    val result = mutableListOf<Byte>()
    withContext(Dispatchers.IO) {
        val outputPath = createFileInFolder("output", fileName, "mp3")
        val fileOutputStream = outputPath.outputStream()
        val chunks = splitTextIntoChunks(translatedText)

        chunks.forEachIndexed { index, chunk ->
            val speechMp3 = textToSpeechChunked(chunk)
            val path = createTempFile("$fileName-chunk-$index", ".mp3")
            path.writeBytes(speechMp3)
            result.addAll(speechMp3.toTypedArray())

            fileOutputStream.write(speechMp3)
            println("Chunk ${index + 1}/${chunks.size} processed")
        }

        fileOutputStream.close()
    }
    return result.toByteArray()
}
```

Generating the RSS Feed

```
fun generateRSS(episodes: List<PodcastEpisode>, outputFile: File): Document {
    val doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().newDocument()

    // Create root rss element
    val rss = doc.createElement("rss")
    rss.setAttribute("version", "2.0")
    doc.appendChild(rss)

    // Create channel element
    val channel = doc.createElement("channel")
    rss.appendChild(channel)

    // Add podcast info
    channel.appendChild(doc.createElement("title").apply { textContent = podcastInfo.title })
    // same for description, link, language, and copyright

    // Add episodes
    episodes.forEach { episode ->
        val item = doc.createElement("item")
        channel.appendChild(item)

        with(doc) {
            item.appendChild(createElement("title").apply { textContent = episode.title })
            item.appendChild(createElement("description").apply { textContent = episode.description })
            // same for pubDate, enclosure, and guid
        }
    }

    return doc
}
```

Example Output

To wrap up, this is the result of translation a part of an English podcast into Russian. Notice the american accent in the Russian speech.

We are also creating an RSS feed behind the scenes for distribution

Example Output

This is Episode 2 of What's in a Name? In the previous episode, we learned how scientists name species, and the controversies that can result from those names. But names don't just matter to scientists; they can impact all of us. In this episode, we're moving out of the universities and scientific publications where names are chosen, and into the public realm, where names chosen by scientists meet non-scientists.

Это второй эпизод "Что в имени?". В предыдущем эпизоде мы узнали, как ученые называют виды и какие могут возникать споры из-за этих названий. Но имена важны не только для ученых; они могут оказывать влияние на всех нас. В этом эпизоде мы уходим из университетов и научных публикаций, где имена выбираются, и переходим в общественную сферу, где имена, выбранные учеными, встречаются с непрофессиональными людьми.

Future Improvements

01 **Automated episode splitting.**

02 **FFmpeg-based merging.**

03 **Custom voice training for TTS.**

Conclusion

Key Takeaways

01 ML deployment is iterative and evolving.

02 Combining OpenAI's tools enables high-quality localization.

03 Future enhancements will further refine the process.

Thank You!