Audio Generation

The script generates audio stimuli for idioms based on the idiom list stored in respective CSV file of Dubrovin's collection. For each idiom, it:

- normalizes the idiom text,
- embeds it into a standardized spoken instruction in Russian/English/German,
- sends this text to Google Cloud Text-to-Speech using a Chirp voice,
- and saves the resulting audio as an MP3 file, one file per idiom.

The script takes idioms from the manually curated file **EN.csv**, which contains the idiom's Russian headword alongside additional information from Dubrovin's English collection. We specified which column should be used for synthesis (e.g., `head_ru`) and allowed for optional indexing (e.g., generating audio only for a subset of idioms).

Each idiom was assigned either:

- its predefined ID from the `EN.csv` file, or
- an automatically generated sequential ID when such a column was absent.

To produce natural and intelligible stimuli, the script applied several normalization steps:

- Idioms written as `word1/word2` were rewritten as `word1 or word2` (same for other languages) so the TTS system would pronounce the alternatives naturally instead of reading the slash aloud.
- Long dashes were standardized, repeated spaces were collapsed, and casing was unified.
- Removal of extraneous whitespace and formatting inconsistencies.

Instead of synthesizing idioms in isolation, we embedded each idiom into a fixed instruction template, where the idiom is introduced and then repeated three times. This ensured consistent prosody and pacing across all audio files.

We used Google's Chirp3 HD Zephyr Russian voice (`ru-RU-Chirp3-HD-Zephyr`), Chirp3 HD Algenib English and German voices (`en-GB-Chirp3-HD-Algenib`, `de-DE-Chirp3-HD-Algenib`)

The script queried the available voices via the API to verify that the desired models existed. If not, warnings were issued but execution continued, allowing for maximal flexibility.

For each idiom:

1. A spoken prompt was constructed using the normalized idiom.
2. The script generated a clean, standardized filename based on the idiom ID and a slugified version of the idiom string.
3. If a corresponding MP3 file already existed in the output directory, synthesis for that idiom was skipped.
4. The text was sent to Google Cloud TTS for synthesis.
5. Successful outputs were written to MP3 files.

The final output consists of a directory of MP3 files, one per idiom, each containing:

- a standardized multi-sentence spoken introduction,
- spoken instances of the idiom itself,
- consistent pacing, pitch, and prosodic settings

Extracting Idioms

After generating long contextual TTS recordings for each idiom, we implemented a second processing stage to automatically extract short audio clips containing only the idiom itself. This step was necessary because the long TTS files contain multiple sentences of instructional context surrounding the idiom. To achieve this, we developed a script that uses OpenAI Whisper for forced alignment and pydub for precise audio cutting and silence trimming.

The script operates on three main inputs:

1. EN.csv – the original Dubrovin's table containing the idiom lemmas and their IDs.
2. Long context audio directory – a folder containing the previously generated long TTS files (one per idiom), named using 4-digit idiom IDs (e.g., `0001_something.mp3`).
3. Output directory – destination for the short idiom-only audio files.

The idiom IDs from `EN.csv` are used to match each idiom to its corresponding audio file.

We load an OpenAI Whisper ASR model to obtain a transcript of the long TTS audio, and word-level timestamps, which are essential for locating where the idiom occurs in the audio.

To reliably match idioms to their spoken occurrences in the transcription, the script performs several preprocessing operations:

- Lowercasing and Unicode-aware normalization of tokens.
- Removal of punctuation and diacritics (except language-specific characters).
- Splitting each idiom into a list of normalized tokens.

For each audio file, Whisper returns a structure containing:

- segments (sentence-like units)
- the words within each segment
- timestamps for every word (`start`, `end` in seconds)

We flatten these into a single list of per-word entries to simplify further processing.

Because idioms may appear multiple times (the long context includes several repetitions) or Whisper may introduce minor tokenization differences, we implemented two complementary alignment strategies:

1. Best contiguous match (primary approach)

The algorithm slides a window across the recognized words and measures how many tokens match the idiom tokens in order. If the match ratio exceeds a threshold (default: 0.6), the corresponding start and end timestamps are accepted.

2. Loose span matching (fallback)

If no contiguous alignment is found, the script collects all word occurrences that match any idiom token and computes: the earliest start time and the latest end time.

If both methods fail, the idiom is marked as “alignment failed”.

Once the correct timestamps are identified:

1. The relevant portion of the MP3 file is extracted using python library pydub.
2. Optional padding is applied (e.g., ±100–300 ms) to avoid cutting off syllables.
3. Leading and trailing silence is automatically removed, leaving ~80 ms of context.
4. The resulting idiom-only clip is saved as:

```
0001_<idiom_slug>_idiom.mp3
```

The script includes several processing safeguards:

- If a target output file already exists, the script skips synthesis for that idiom.
- Missing long-context audio files are logged as failures.
- Whisper transcription errors, timestamp failures, or audio-cutting exceptions are recorded but do not interrupt execution.

At the end of the run, a summary of successfully extracted idioms, skipped items and failures is displayed.

To maximize coverage, the script includes a second pass that attempts to recover idioms that failed in the first round.

This retry stage re-transcribes the audio, allows a lower match threshold (e.g., 0.4), repeats the alignment and audio-cutting logic. If alignment still fails, the idiom is permanently marked as unsliceable. For these remaining items, we manually extracted idiom-only segments using QuickTime Player to obtain full dataset coverage.