Song Search using CLAP

SUSE Hack Week 25

Goals

Search music using natural language descriptions

- Find songs by describing what you hear: "piano melody", "female vocalist"
- Compare songs to find similar tracks
- Experiment AI with Copilot Code Assistant

CLAP Overview

CLAP (Contrastive Language-Audio Pretraining) - Bridging audio and language

Core Concept:
CLAP can represent both songs and text descriptions as **512-dimensional vectors** in the same mathematical space. This enables searching for music using natural language queries.

How it works:

- A song becomes a 512-number vector capturing its audio characteristics
- A text query like "piano music" also becomes a 512-number vector
- Vectors that point in similar directions represent similar content
- We measure similarity to find songs matching the description

Model: `music_audioset_epoch_15_esc_90.14.pt` - Specialized for music (71% genre accuracy, trained on ~4M samples)

Implementation

Two Modes:

- **Single Analysis:** `single_analysis.py` - One song vs one query
- **Batch Analysis:** `multiple_analysis.py` - All songs vs all queries

Architecture:

- `clap_analysis.py` - Core library (shared)
- Automatic model download & caching
- Parallel processing across CPU cores
- CSV output with detailed metrics

Cosine Similarity Scores:

- Range: -1 (opposite) to $+1$ (identical)
- Thresholds: > 0.3 = HIGH, > 0.15 = MODERATE, ≤ 0.15 = LOW

Query Wording Matters

Discovery: Small word changes can dramatically affect results!

Example - "Paint it Green" by SUSE Band (female vocalist rock):

Query	Score	Match
"female vocalist"	0.328	HIGH
"female voice"	0.062	LOW
Difference	-0.267	**5× worse!**

Why? The model was trained on music descriptions. Professional terminology like "vocalist" appeared more frequently than casual words like "voice".

Tip: Use music industry vocabulary for better results.

Finding Similar Songs

Bonus Feature: Beyond text search, we can find which songs sound similar to each other.

How It Works:

- Each song gets its own 512-dimensional vector representation
- We compare these vectors between all song pairs
- Songs with similar vectors have similar audio characteristics
- Same similarity metric as text queries

Advantage: No additional processing needed!

- Song vectors are already computed during text query analysis
- Comparison is instant (~0.001 seconds for 10 songs)
- Can help discover related songs in your collection

Use Cases:

- Find songs with similar instrumentation or mood
- Discover patterns in your music collection
- Create automatic playlists based on audio similarity

AI Support in Development - What Went Well

Project developed with GitHub Copilot (Claude Sonnet 4.5)

1. **Research & Model Selection**
   - Compared CLAP models → selected `music_audioset` (71% GTZAN accuracy)
   - Read CLAP paper, explained architecture accessibly
2. **Code Analysis & Technical Discovery**
   - Examined checkpoint: model lacks fusion, max 10s segments
   - Designed overlapping segment solution
3. **Data Analysis & Documentation**
   - Found "vocalist" vs "voice" pattern (5x difference in results)
   - Generated all technical docs with examples

AI Support in Development - What Needs Improvement

1. **Unrequested Code Changes**
   - Single-file CLI disappeared, score icons changed without asking
   - **Problem:** Full file rewrites hide unintended changes
   - **Lesson:** Always review diffs carefully
2. **First Solution $\neq$ Best Solution**
   - **Performance:** Song analyzed per query $\rightarrow$ challenged $\rightarrow$ **15x faster**
   - **Threading:** Single-threaded $\rightarrow$ asked "faster?" $\rightarrow$ multiprocessing

Key Lesson: Working $\neq$ optimal. Challenge AI with domain knowledge.

Best Practices for AI-Assisted Development

Do's:

- Ask AI to research and compare options (models, approaches)
- Request code analysis to understand existing implementations
- Share actual data/errors for pattern recognition
- Challenge solutions: "Can this be faster?" "Is there a better way?"
- Request documentation with examples from your project

Don'ts:

- Don't accept first working solution as optimal
- Don't let AI rewrite entire files without showing diffs
- Don't assume AI made only requested changes - review carefully
- Don't trust performance without testing alternatives

Bottom line: AI is a powerful assistant for research, analysis, and documentation. For code: it provides a working starting point, but optimization requires your expertise and critical thinking.

References

- **CLAP:** https://github.com/LAION-AI/CLAP - The main model being researched
- **Hugging Face:** https://huggingface.co/docs/transformers/model_doc/clap - Pre-trained models for CLAP
- **Free Music Archive:** https://freemusicarchive.org/ - Creative Commons songs for testing
- **SUSE Hack Week 25 Project:** https://hackweek.opensuse.org/25/projects/clap-machine-learning-to-search-song-starting-from-text - Project page
- **Project Repository:** https://github.com/gcolangiulisuse/hw25-song-search - GitHub repo of the project