**Portfolio Milestone: Dynamic Time Warping App**

Chioma Chance

Colorado State University Global

CSC506: Design and Analysis of Algorithms

Dr. Banerjee

02/23/2025

**Dynamic Time Warping App for Music Sample Identification**

**Abstract**

Music production and sampling have become increasingly sophisticated, making it essential to develop tools that can accurately identify sampled tracks despite variations in pitch, tempo, or arrangement. This project focuses on building an application that utilizes **Dynamic Time Warping (DTW)** to compare audio files and determine sample similarities, even when the audio has been altered. The implementation will begin with **Flask** for its simplicity and ease of deployment, with plans to migrate to **FastAPI** in the future for improved performance and scalability.

**Introduction**

The ability to detect music samples in modified tracks is crucial for producers, music analysts, and copyright enforcement agencies. Traditional fingerprinting methods struggle when samples are altered slightly, such as through time-stretching or pitch-shifting. **Dynamic Time Warping (DTW)** is a powerful algorithm that can align sequences with non-linear distortions, making it well-suited for music sample identification. This project will implement a DTW-based approach to compare music files and match samples despite changes.

**Project Scope and Approach**

The application will allow users to upload an audio track and compare it against a database of existing tracks to detect potential matches. Since raw audio files can be large and computationally expensive to process, the system will first convert the audio into a more **DTW-compatible format**, such as **Mel-Frequency Cepstral Coefficients (MFCCs)** or **chromagrams**. This will optimize pattern recognition and minimize processing overhead.

**Phase 1: Research and Data Processing**

1. **Understanding DTW**: Reviewing literature on DTW in music retrieval and testing different implementations.
2. **Feature Extraction**: Determining the best audio features to extract (MFCCs, chromagrams, or spectral features) to maximize DTW efficiency.
3. **Data Storage**: Deciding on a format for storing processed music samples, ensuring efficient retrieval.

**Phase 2: Backend Development with Flask**

1. **API Development**: Implementing a simple REST API using Flask to handle music uploads and comparisons.
2. **Audio Processing Pipeline**: Building a function to convert uploaded music into feature representations before applying DTW.
3. **Similarity Computation**: Running DTW on processed audio data to compute similarity scores.
4. **Database Integration**: Storing indexed audio features to speed up comparison.

**Phase 3: Frontend and User Experience**

1. **User Interface**: Designing an intuitive UI for users to upload and compare tracks.
2. **File Handling**: Exploring user-friendly upload methods, potentially supporting **YouTube-to-MP3 conversion** for easier sample submission.
3. **Visualization**: Providing graphical feedback on similarity scores and matched audio segments.

**Phase 4: Deployment and Initial Testing**

1. **Deploying Flask Application**: Hosting the Flask-based app on a cloud service (e.g., AWS, Heroku).
2. **Testing and Optimization**: Running sample queries to evaluate performance and adjust thresholds.
3. **Collecting User Feedback**: Refining based on usability testing.

**Future Plan: Transition to FastAPI** While Flask is an excellent framework for development and initial deployment, **FastAPI** offers superior performance due to its asynchronous capabilities and automatic OpenAPI documentation. As the app gains traction and scalability becomes crucial, the migration to FastAPI will enable:

* **Faster request handling** with async processing.
* **Improved scalability** for handling multiple user queries concurrently.
* **Automatic API documentation** with built-in OpenAPI and Swagger support.

**Conclusion**

This project will create a **user-friendly music sample detection app** that leverages **Dynamic Time Warping (DTW)** for accurate matching despite alterations. The initial development will use Flask due to its ease of setup, followed by a **transition to FastAPI** for better scalability. Research into **feature extraction techniques, database storage, and frontend usability** will ensure a functional and efficient tool for **music producers, analysts, and copyright investigators**.

**References**

1. GeeksforGeeks. (n.d.). Dynamic Time Warping (DTW) in Time Series. Retrieved from <https://www.geeksforgeeks.org/dynamic-time-warping-dtw-in-time-series/>
2. GitHub. (n.d.). DTW Python Implementation. Retrieved from <https://github.com/DynamicTimeWarping/dtw-python/blob/master/dtw/__main__.py>
3. ArXiv. (n.d.). Dynamic Time Warping Applications in Music Analysis. Retrieved from <https://arxiv.org/abs/2310.18128>
4. ResearchGate. (n.d.). Benchmarking DTW for Music Retrieval. Retrieved from <https://www.researchgate.net/publication/221410471_Benchmarking_Dynamic_Time_Warping_for_Music_Retrieval>