# 194.166 Einführung in Information Retrieval 2025W
# Exercise 2: Music Retrieval/Audio Identification

Milestone 2.2: Fingerprint Matching

Exercise 2 deals with a specific music retrieval task, namely audio identification. Milestone 2.2 covers fingerprint generation and matching strategies.

## Material

This milestone is (again) based on the original Shazam background paper by Avery Wang "**An Industrial Strength Audio Search Algorithm**" (ISMIR 2003), **Chapter 7: Content-Based Audio Retrieval** in **Fundamentals of Music Processing** (2nd ed., Springer, 2021) by Meinard Müller, and, optionally, an implementation of the **MinHash algorithm in the context of audio identification** by Wishnek and Lysandrou as a student project at the University of Colorado Boulder.

## Task 1: Hash Generation and Storage (30%)

Use the best parameter configuration for peak detection as identified in M2.1. Rather than matching segments of constellation maps, this task deals with the generation of hash values over the full length of audio tracks. To this end, follow the strategy outlined in the paper by Wang, to create pairs of anchor points and construct hashes containing the frequencies and time difference of these anchor points.
Find suitable resolutions to balance the tradeoff between precision and storage requirements for both frequency and time resolution with the goal of representing them using a 32-bit unsigned integer as hashes as indicated by Wang. Regarding choosing the time resolution representation, keep this in mind also when defining the parameters of the target zone, which further has impact also on the number of hashes created.

- To this end, define four different target zone areas (as combinations of two distinct frequency ranges and two distinct time windows chosen) for comparison of their effect on the performance in terms of retrieval efficacy, storage complexity, and retrieval times (see task 2).
- Optional: consider using an R-Tree for spatial representation and faster identification of candidates
- Optional: instead of strictly defining a target zone, consider using a fixed number of target candidates per anchor point (within constraints of time). Choose four different fixed numbers of candidates for comparison (or mix with the target zone approach to end up with in total four different configurations for comparison).

Document the number of hashes generated for the database (in total and per track on average) and the storage requirement depending on the chosen parameters for each configuration considered.

Choose an efficient storage mechanism for the created hashes. This can range from a simple lookup table to a hashtable to an inverted index (or a combination of different elements thereof). Make sure to store the information needed to identify the associated track ID and time offset of the occurrence hash (if needed). For choosing or building a strategy, consider the following three examples of hash storing (and corresponding matching) for reference:

1. “Database” approach as outlined by Wang: Find all matching hashes in the database (or used structure), sort (or assign) according to track ID and pair occurrence timing offsets of reference and query clip.
2. Inverted Index as described by Müller: storing the information needed for matching using the generated hashes; note that the described index focuses on efficiency of comparison of constellation maps, nonetheless, the concept of using the hashes as keys to (sorted) lists of tracks and timings could apply here as well.
3. MinHash set comparison approach as implemented by Wishnek and Lysandrou as a fast approach to comparing sets of “shingles”/hashes

Discuss the data structure built for storage and lookup and its implications for the matching process. Report on the memory/storage requirements of the built data structure.

## Task 2: Audio Identification (30%)

Repeat the retrieval experiments of Milestone 2.1, this time using the hash representation and storage. Use a matching strategy corresponding to your chosen data structure and reference approach. Compare retrieval performance (efficacy, query times) for all four configurations and all types of queries individually and combined.

## Task 3: Scale-Up (30%)

Scale up the best performing configuration of creating hashes to the full length of tracks (not only the first 30 seconds). Scale up the database by adding further tarballs from the MTG-Jamendo dataset (see M2.1 task 1). Every added 5 tarballs account for 5%-points of this task. Report on performance with increasing database size.

## Task 4: Reporting and Submission (10%)

Submit the full (adapted) pipeline including the overlapping parts from M2.1. In addition to submitting your generated queries and documented code, write a report detailing your findings and design choices, including visualizations, figures, and tables. In case you are submitting a (very well!) documented and illustrated Python notebook, this report can consist of an html export of the notebook. In any case, make sure to include discussions on design choices where they had to be taken. Submit all files in a single zip archive on TUWEL.