# Cover Songs Retrieval and Identification

PhD collaboration with the Center for Digital Music

Julien OSMALSKYJ

University of Liège
Montefiore Institute
Belgium

josmalsky@ulg.ac.be

www.montefiore.ulg.ac.be/~josmalskyj

January 12, 2015

# Contents

# Contents

# Introduction

## Initial goal

Identify a live (unknown) performance track, that is, find any information related to the original track. The unknown track is usually a different version of the original track.

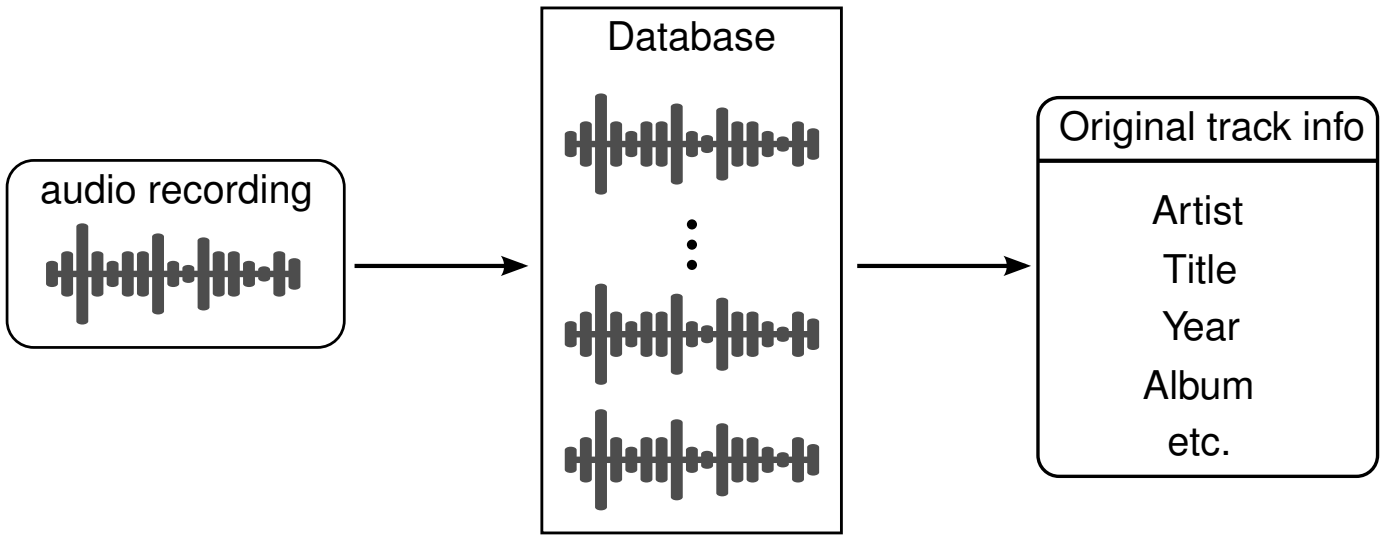



FIGURE: General goal

# Cover Songs

The problem of identifying an unknown track is related to the problem of *cover songs recognition*.

## Definition

A cover song is a possibly unknown version of an existing musical track, that can differ in tempo, pitch, instrumentation, timbre and other parameters.

Because of these differences, finding covers automatically is not an easy problem.

# Applications

Cover songs recognition could be useful for a wide range of applications :

- plagiarism detection
- live music recognition
- browsing music collections
- query-by-example
- discovering music through covers
- etc.

# Search Engine

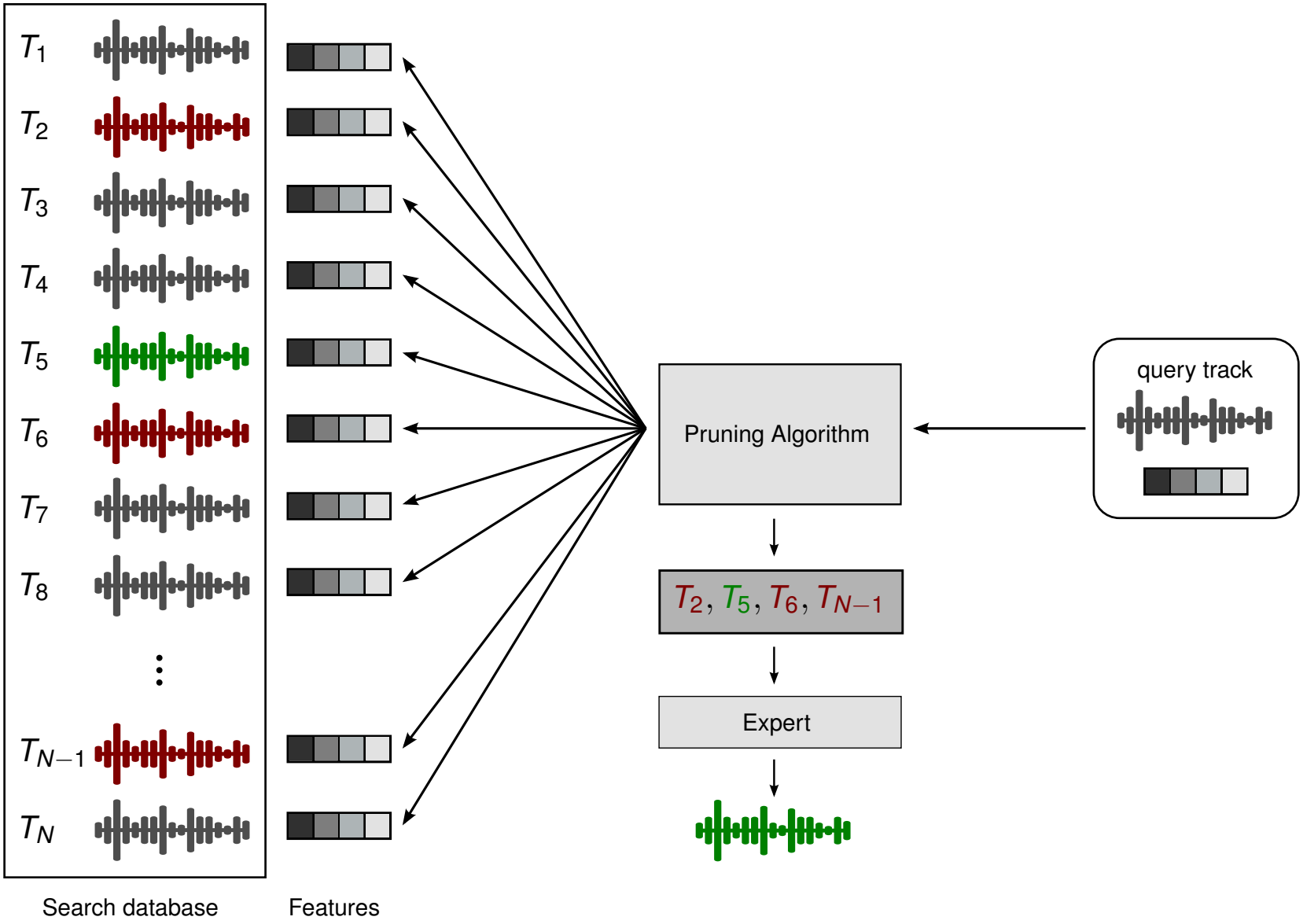
A cover song recognition system is an *Information Retrieval System* (IRS) relying internally on a *Content Based Retrieval System* (CBRS).

For an audio query $q$, features are computed and used by the CBRS to identify samples in a database that are related to the $q$.

A search engine can be implemented in two steps :

- A pruning algorithm that returns only samples related to the query $q$.
- An expert that predicts the requested information based on the pruned set.

# Search Engine



# Contents

# Definitions

- Let $O$ be the space of input objects (tracks, sounds, etc.) ;
- Let $I$ be the space of output information (author, album, etc.) ;

## Definition

$f : O \to I$ is a function that associates an object to an information.

## Definition

$s : O \times O \to \mathcal{B}$ is a similarity function such that $s(o_1, o_2) = T \Leftrightarrow f(o_1) = f(o_2)$, where $\mathcal{B} = \{T, F\}$ is the set of booleans and $o_1, o_2 \in O$.

# Similarity function

An IRS aims at defining a function $\widehat{f}$ as close as possible to $f$.

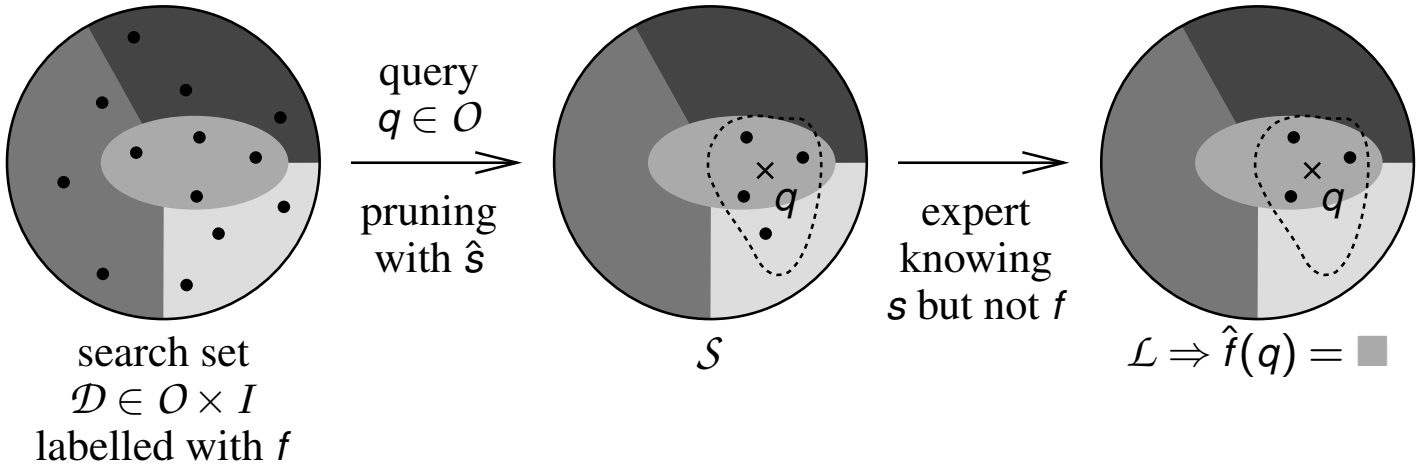
A CBRS defines a function $\widehat{s}$ that approximates $s$.

In practice, $s$ can only be approximated by a function $\widehat{s}$.

An *expert* knowing $s$ but not $f$ is often available (even human expert).

The *pruning* is based on the function $\widehat{s}$.

# IRS based on a CBRS I

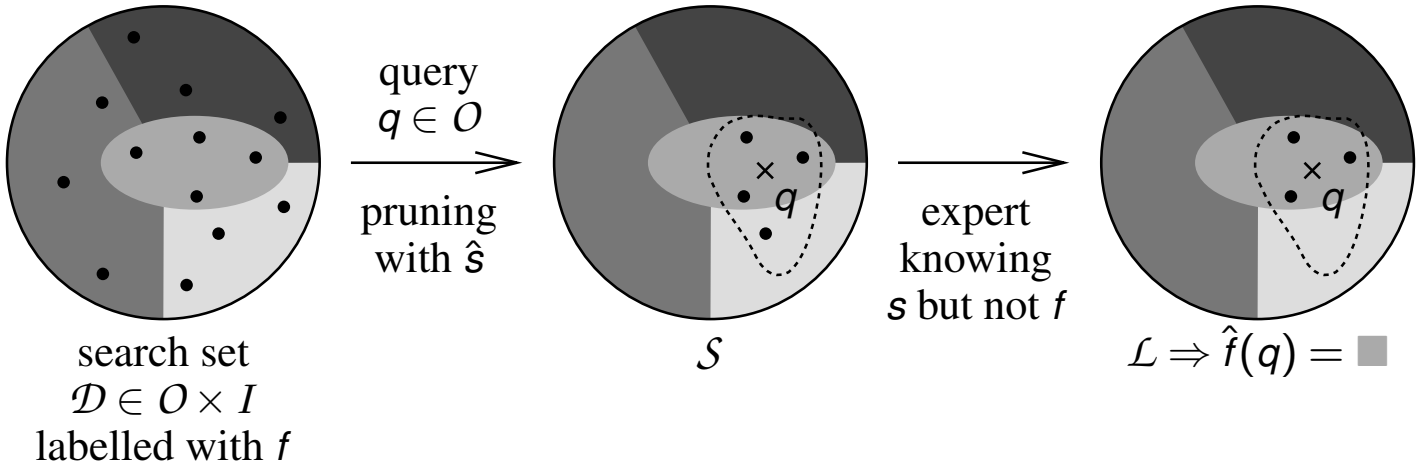



A search database $\mathcal{D}$ is defined as a set of samples drawn from $O \times I$, with the samples $(o, i) \in \mathcal{D}$ such that $f(o) = i$.

For a query $q$, the CBRS prunes the database and returns the subset

$$\mathcal{S} = \{(o, i) | (o, i) \in \mathcal{D} \wedge \widehat{s}(o, q) = T\}$$

# IRS based on a CBRS II



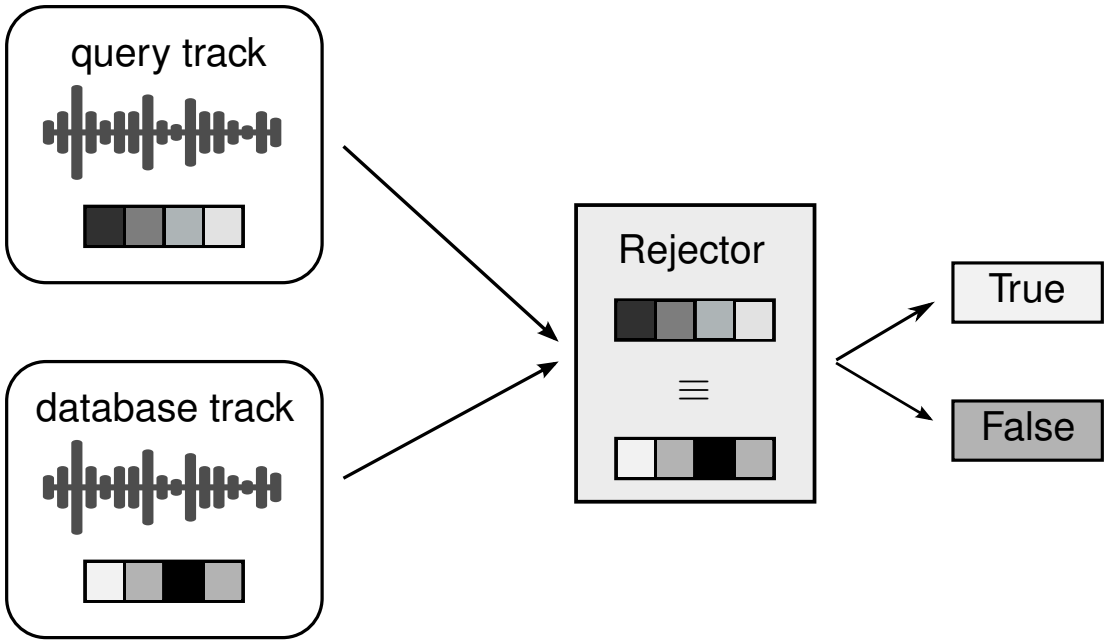

Then, an expert knowing $s$ finishes the identification. It returns the subset

$$\mathcal{L} = \{(o,i) | (o,i) \in \mathcal{S} \wedge s(o,q) = T\}$$

If $|\mathcal{L}| \neq 0$, it returns $\widehat{f}(q) = i$, where $(o,i)$ is any element in $\mathcal{L}$.

# Rejector

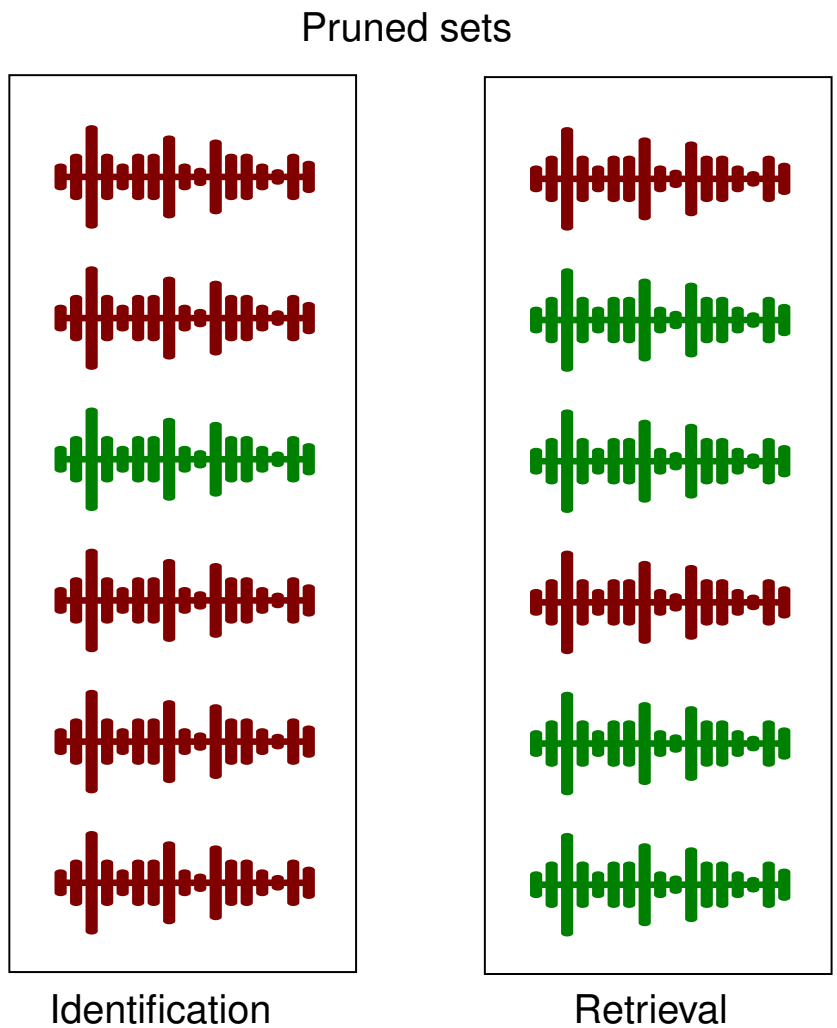
The estimated similarity function $\widehat{s}$ used for pruning has been called a rejector.




A rejector takes two tracks as an input and returns whether it considers them similar or not.

# Two possible situations

- Identification : at least one match in the pruned set.
- Retrieval : as many matches as possible.



# This PhD

Focus mainly on the first case : identification of cover songs.

- Work on rejectors
  - Analysis of performances.
  - Rejectors mainly with machine learning.
  - Combination of rejectors : many schemes studied.
- Evaluation
  - Work towards a standard evaluation method.
  - Design of an evaluation space : Prune-Loss.
  - Evaluation with standard large dataset.
  - Development of an evaluation framework : CAROLYNE.

# Contents

# Two Techniques

Cover songs have been widely studied in the litterature.

Two main methods used by researchers :

- Index-based methods

  Extract features, build an inverted file.

- Comparison-based methods

  Compare query to entire database.

Several references for both cases given below. Many more exist.

# Index-based Methods

Methods based on an inverted file, similar to text retrieval systems.

- Casey et. al., 2007 [2] : using Locally Sensitive Hashing (LSH) with audio shingles.
- Kurth et. al., 2008 [6] : codebook with CENS chroma vectors.
- Lu et. al., 2012 [7] : codebook of chroma features with KMeans.
- Other methods such as Shazam or SoundHound : not suitable for cover songs.

# Comparison-based Methods

Methods based on an exhaustive comparison of the query with the tracks of a database.

- Ellis et. al., 2007 [3] : beat-synchronized chromas.
- Kim et. al., 2008 [5] : chromas and deltas between chromas.
- Grosche, 2012 [4] : large audio shingles.
- Bertin-Mahieux, 2013 [1] : cover songs with a large-scale database.
- Van Baelen et. al., 2014 [10] : mid and high level music analysis.

# Problems

Cover songs recognition is still an active field of research.

- No ideal system yet in a large-scale setup.
- Difficult to compare results : No common dataset, no common metrics, no common evaluation method.
- Huge lack of evaluation databases : biggest one is the Second Hand Song Dataset.
- Not so much machine learning in the litterature.

Still a lot of work to do !

# Contents

# Experimental setup

We need a large search database to design effective and efficient rejectors.
Lack of large-scale database for MIR in general :

| Dataset | Songs / samples | Audio available |
|---|---|---|
| Cover80 | 80 | Yes |
| RWC | 465 | Yes |
| Bimbot et al. | 500 | No |
| CAL-500 | 502 | No |
| GZTAN genre | 1,000 | Yes |
| Billboard | 1,000 | No |
| MusiCLEF | 1,355 | Yes |
| SALAMI | 1,400 | No |
| USPOP | 8,752 | No |
| CAL-10K | 10,870 | No |
| Magnatagatune | 25,863 | Yes |
| **Second Hand Song Dataset (SHSD)** | **18,196** | **No** |
| **Million Song Dataset (MSD)** | **1,000,000** | **No** |

TABLE: List of standard datasets for MIR tasks

# Million Song Dataset

## Definition

The Million Song Dataset (MSD) is a freely-available collection of *audio features and metadata* for a million contemporary popular music tracks.

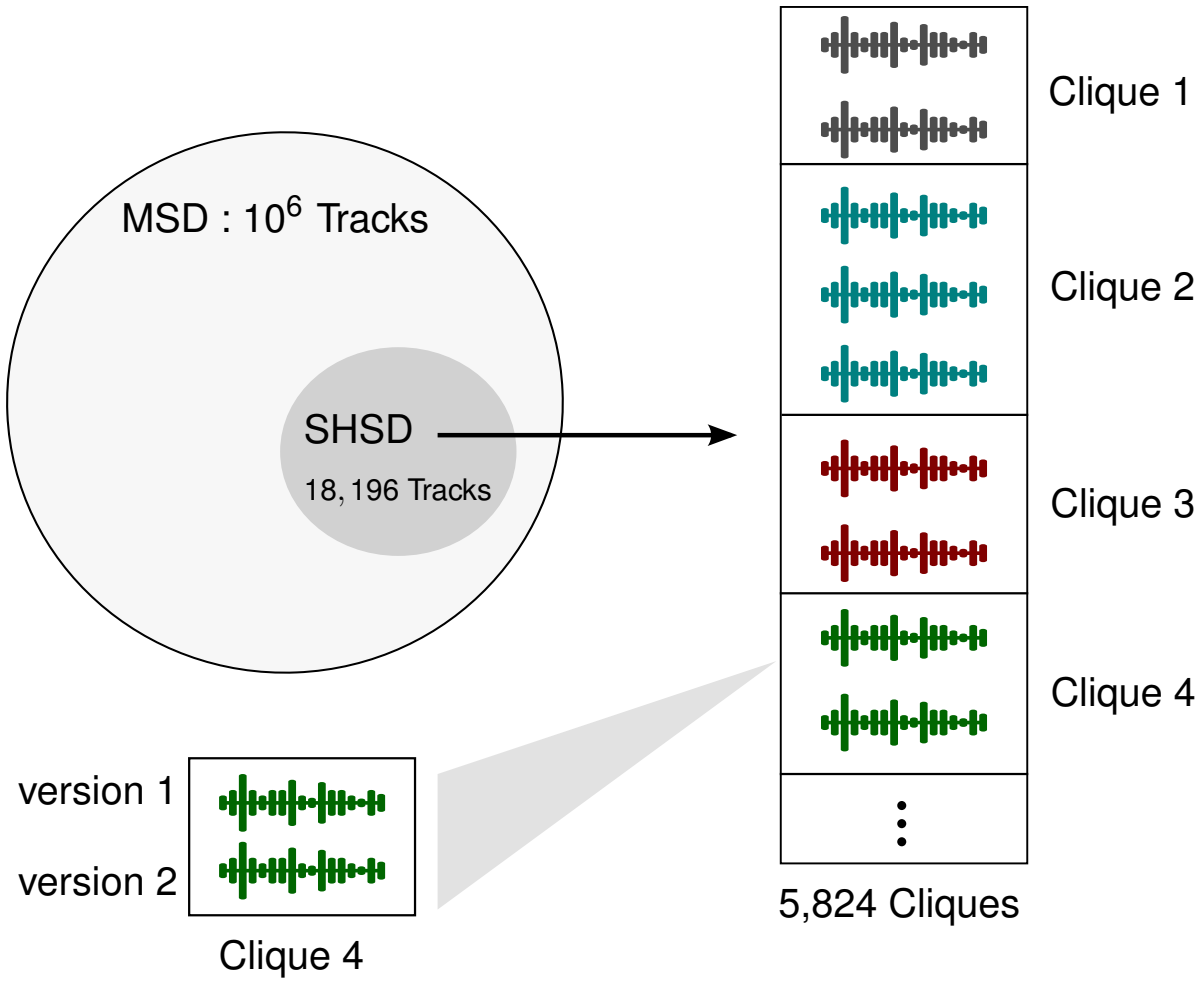
The Second Hand Song Dataset (SHSD) is a subset of the MSD.

The SHSD is organized in 5,824 cliques of cover songs.

Tracks within a clique are different versions of the same song.

Unfortunately, no audio data is available with the dataset.

# Million Song Dataset



# MSD Features

Rejectors can only use *pre-computed features* from the MSD.
Many features are provided with the MSD.

| Feature | Type | Descripttion |
|---|---|---|
| energy | float | energy from listener point of view |
| key | int | key the song is in |
| key confidence | float | confidence measure |
| loudness | float | overall loudness in dB |
| mode | int | major or minor |
| mode confidence | float | confidence measure |
| segments pitches | 2D array float | chroma feature, one value per note |
| segments start | array float | musical events, note onsets |
| segments timbre | 2D array float | texture features (MFCC+PCA-like) |
| ... | ... | ... |

TABLE: Examples of features in the Million Song Dataset

# Two types of rejectors

Rejectors can be based on a distance or a probability.

For a distance, two tracks are considered similar when the distance between both tracks is small.

For a probability, two tracks are considered similar when the probability of similarity is high.

Probability based rejectors are built using machine learning algorithms, typically, random forests.

# Build probabilistic rejector

Probabilistic rejectors use a machine learning algorithm to learn a similarity measure.

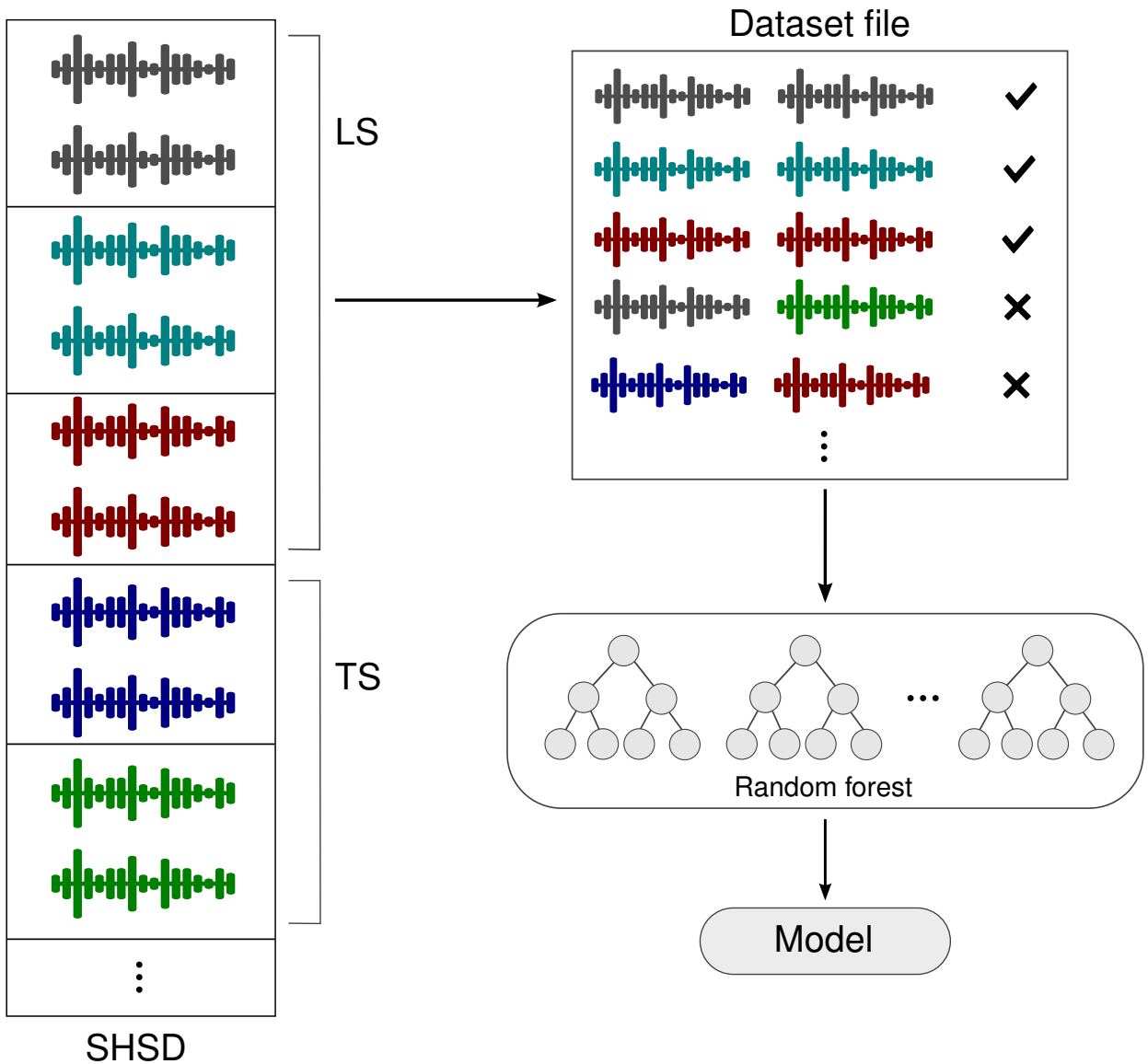
SHSD is split in two parts : a Learning Set (LS) and a Test Set (TS).

Tracks of the LS are used to build a dataset file :

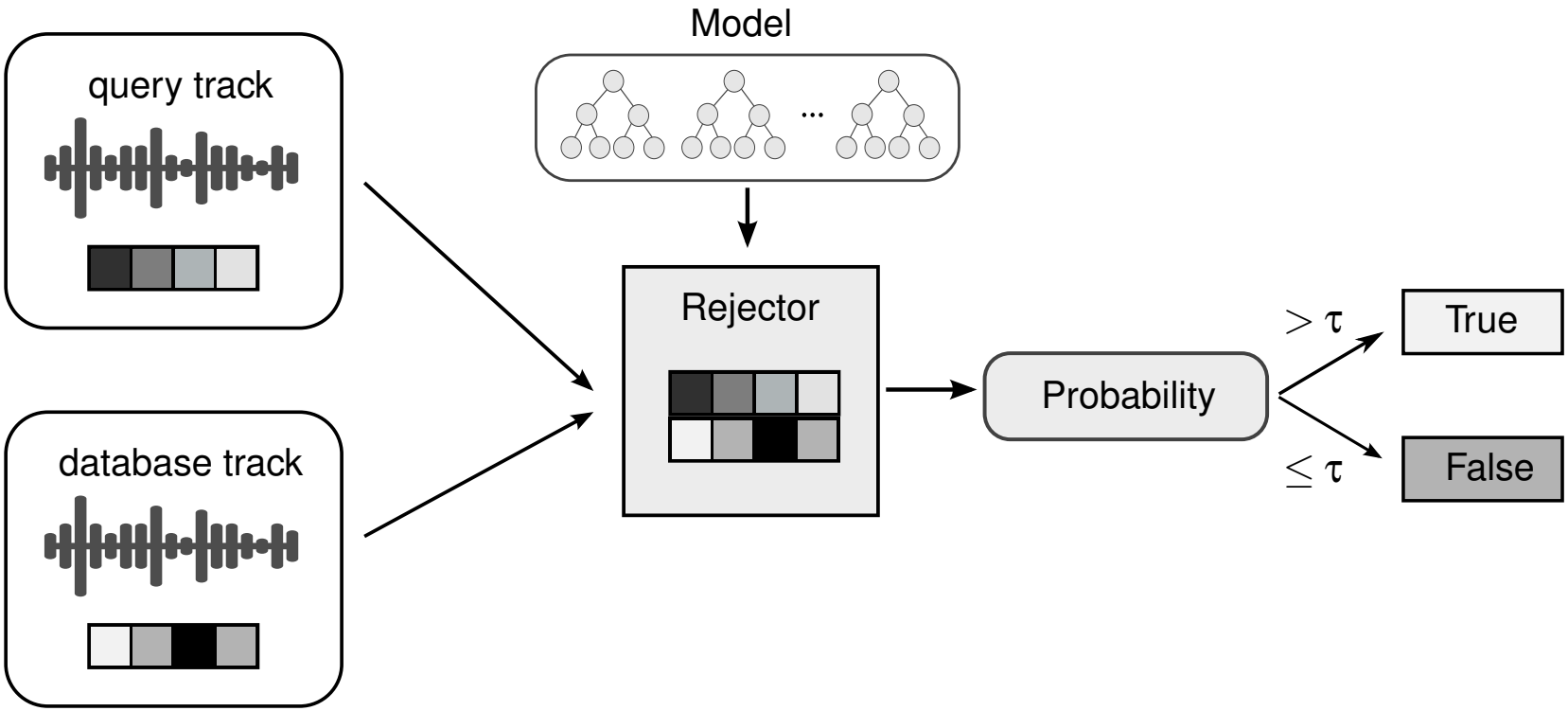
- Two tracks of the same clique are labelled similar.
- Two tracks of two different cliques are labelled dissimilar.

The LS dataset is sent to the learning algorithm, for example a Random Forest.

# Build probabilistic rejector



# Use probabilistic rejector




The rejector uses the machine learning model to compute a probability of similarity. The probability is compared to a threshold $\tau$.

# Single Rejectors I

Various rejectors were implemented for several features, including low-level and mid-level features.

Low-level features :

- compactness
- zero-crossings
- spectral flux, rolloff, variability and centroid
- root mean square (RMS)
- tempo
- duration
- number of beats

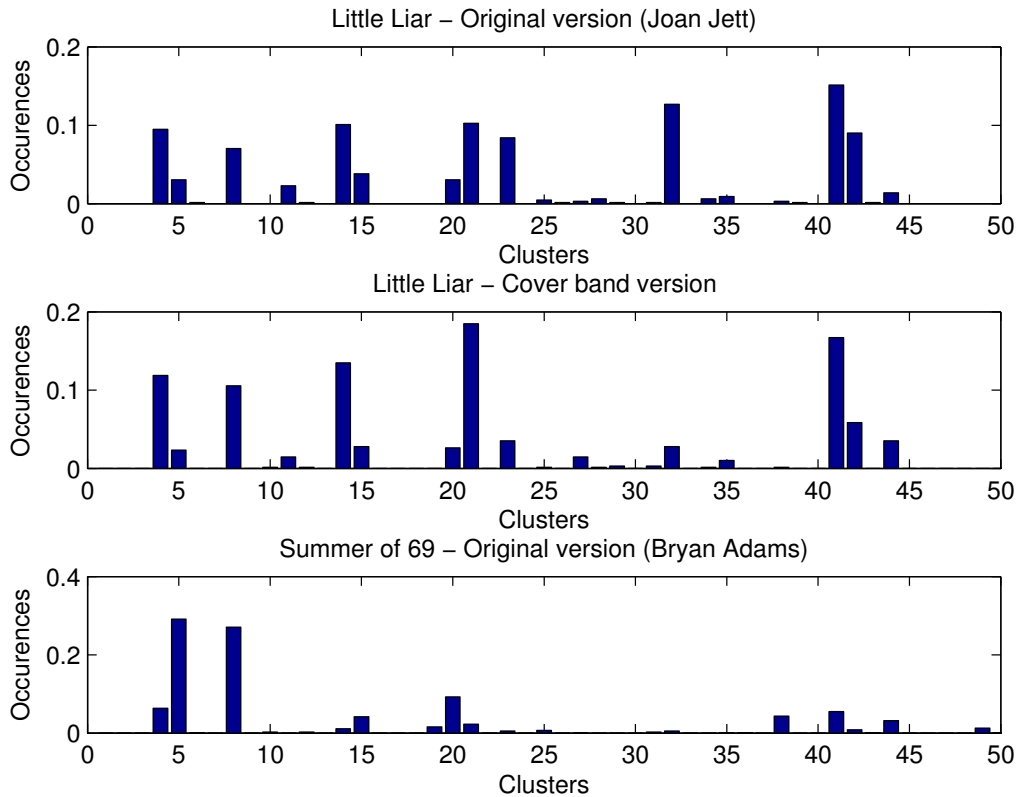
Low-level features studied in order to combine them later.

# Single Rejectors II

Mid-level features :

- chroma-based features
- MFCCs-based features

Example : Bag-of-words of chroma features [8].



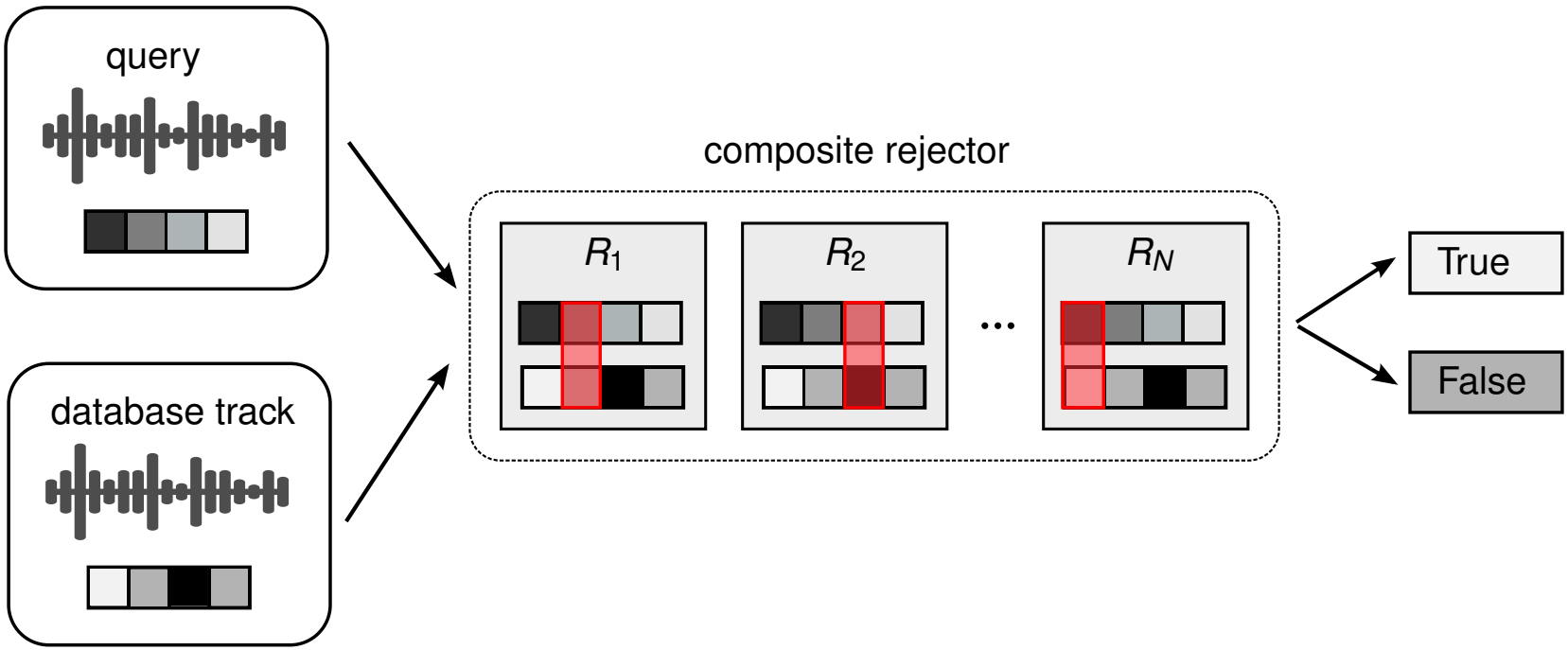
# Composite Rejectors

Combine single rejectors into a composite rejector.

Several ways of combining rejectors :

- Boolean combination : Union, Intersection, Majority vote [8]
- Probabilistic combination : sum, product, etc. [9]
- Machine learning combination : model grouping many features.
- Combination in the Evaluation Space, e.g. ROC space : e.g. IBCAll technique

# Composite Rejectors




Each single rejector $R_i$ takes its decision based on a different feature.

# Contents

# Evaluation

Reminder : No large-scale database with audio-data available.

- Researchers often use personal collections.
- It makes the results difficult to compare.
- Very few databases organized in *groups* of cover songs.

As stated before, the only solution for now to develop scalable systems is the MSD + the SHSD.

We need a common database, common metrics, and a common evaluation algorithm to compare systems.

# Metrics

Most research is about retrieving as many matches as possible for a given query.

Therefore, metrics are used according to that goal : *Mean Average Precision (MAP), Average Rank, Number of covers in top 10, Precision-Recall, Accuracy*.

Flaws :

- If database organized in cliques, we do not need *all matches*.
- With poor performance, is it useful to know that first cover is at position e.g. 300,000 ?

# Introducing Prune-Loss space

For evaluating search engines for cover songs using a *pruning algorithm*, it makes sense to plot the pruning rate against the loss rate.

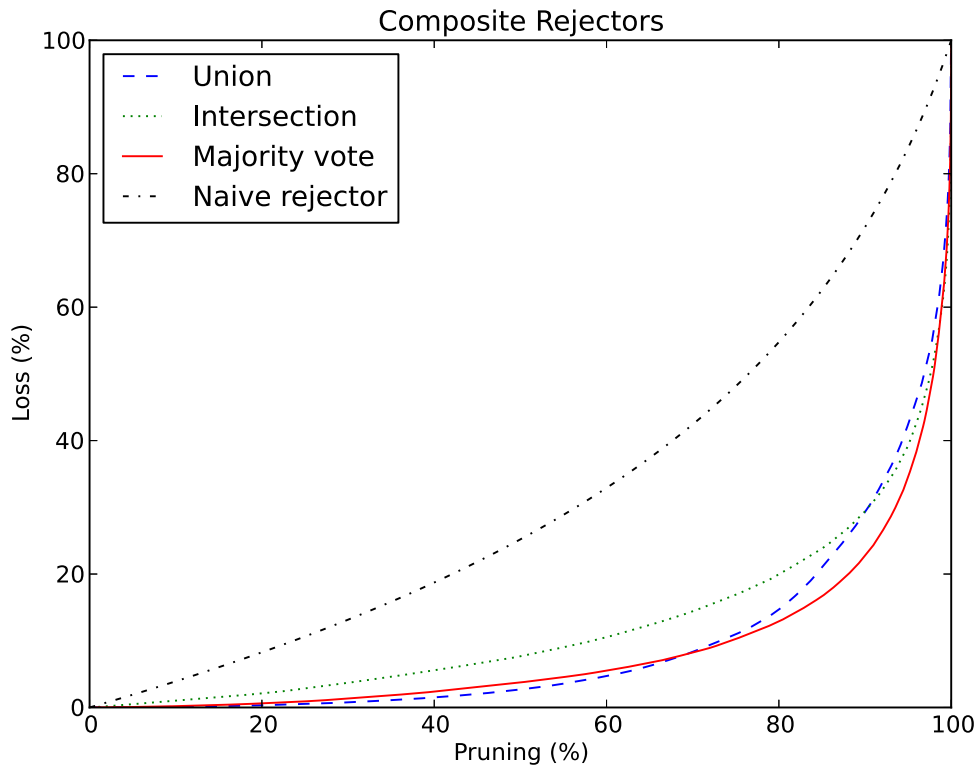



FIGURE: Example of Prune-Loss curve

# Loss rate

With an evaluation database organized in *cliques* :

## Definition

The loss rate is the mathematical expectation, over all possible queries, of the probability to discard all matching samples in $\mathcal{D}$.

For an entire database, we can show that

$$\text{loss} = \frac{1}{N} \sum_{i=1}^{N} \underbrace{p\left[\hat{s} = 0 | s = 1\right]}_{\text{FNR}}^{|C_i|-1}$$

Where $|C_i|$ is the size of the clique $i$, $N$ is the number of cliques.

# Pruning rate

## Definition

The pruning rate is the mathematical expectation, over all possible queries, of the proportion of irrelevant samples from a database $\mathcal{D}$ that are pruned.

$$\text{prune} = \underbrace{p(\widehat{s} = F | s = F)}_{\text{TNR}}$$

For a pruning algorithm, what matters is to have a high pruning rate with a minimum loss rate.

# Fast evaluation algorithm

Pre-compute thresholds and evaluate prune, loss and any other metrics for all thresholds directly.

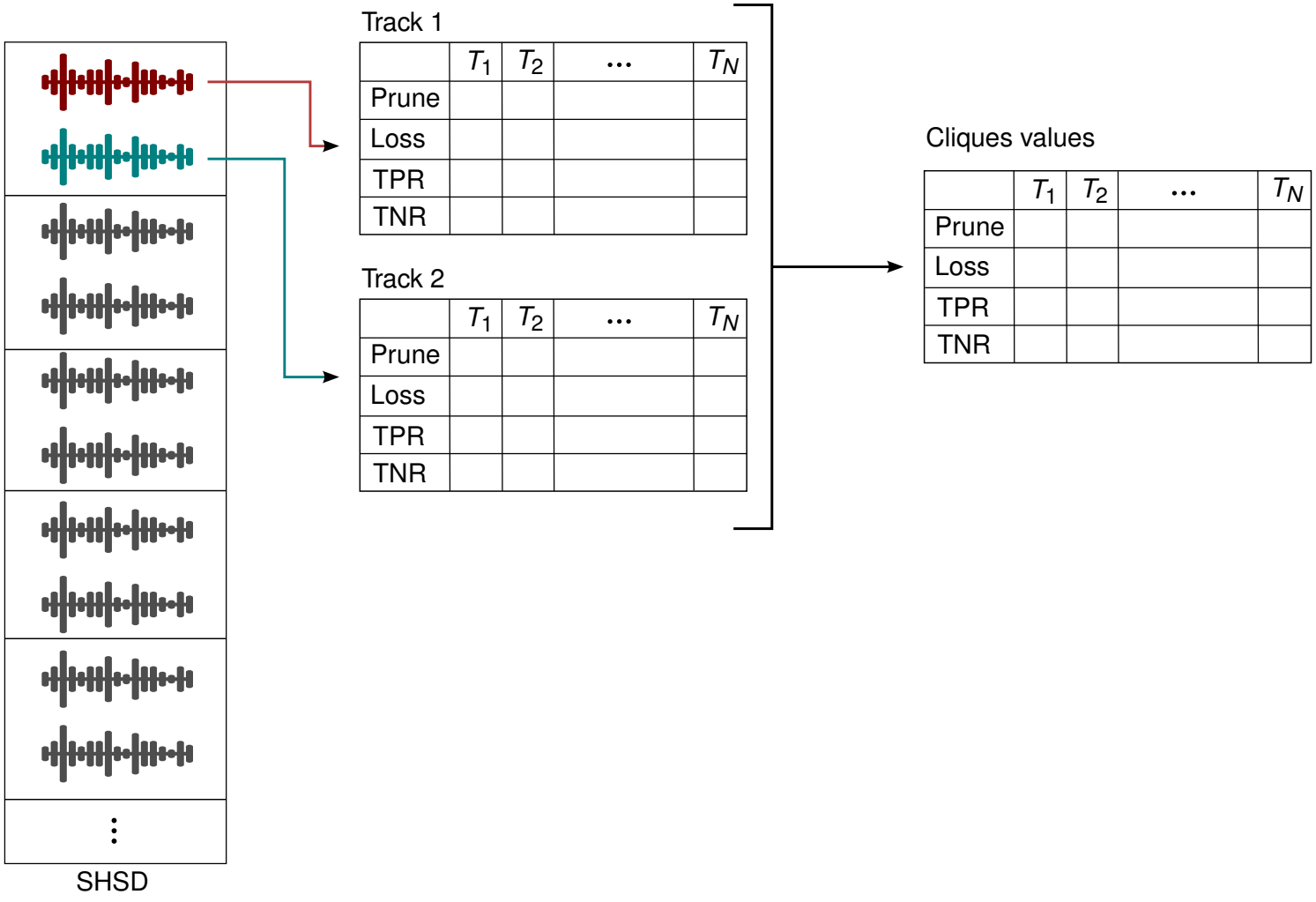
For thresholds selection, compute $N$ similarity values using a rejector and sort them.

Compute *per track* values and store them.

Compute clique values by averaging tracks values by the number of items in a clique.

Finally, compute global values by averaging by total number of cliques.

# Fast evaluation algorithm




| | $T_1$ | $T_2$ | ... | $T_N$ |
|---|---|---|---|---|
| Prune | | | | |
| Loss | | | | |
| TPR | | | | |
| TNR | | | | |

FIGURE: Evaluation algorithm

# Evaluation Framework

All the above-mentioned considerations have been implemented in an Evaluation Framework Software.

The framework is a first attempt toward a standard evaluation method for cover songs identification and retrieval.

Software written in C++ for fast processing.

It is still a work in progress : 6,984 lines of code.

Designed to be easily improved : new metrics, new evaluators, new learning algorithms, etc.

Name of the framework : CAROLYNE.

# Evaluation Framework : CAROLYNE

- Framework entirely written in C++ for fast processing.
- Oriented-object code with design patterns (builders, loaders, etc.).
- Full integration with the MSD and the SHSD.
- Easy to build and experiment new rejectors.
- Easy to combine rejectors using several combination schemes.
- Allows to use machine learning to build rejectors models.
- Outputs results using common metrics and ROC, PR and PL spaces.

# Contents

# Experiments

Several experiments made with the framework that led to publications.

- Analysis of simple rejectors and their combination [8]
- Analysis of low-level features using various combination schemes for cover songs [9]
- Combination of classifiers in the ROC space and application to the PL space.

# Boolean combination of rejectors

Combination of rejectors based on tempo, duration, beats and chromas. Introduction of the bag-of-words of chroma features.

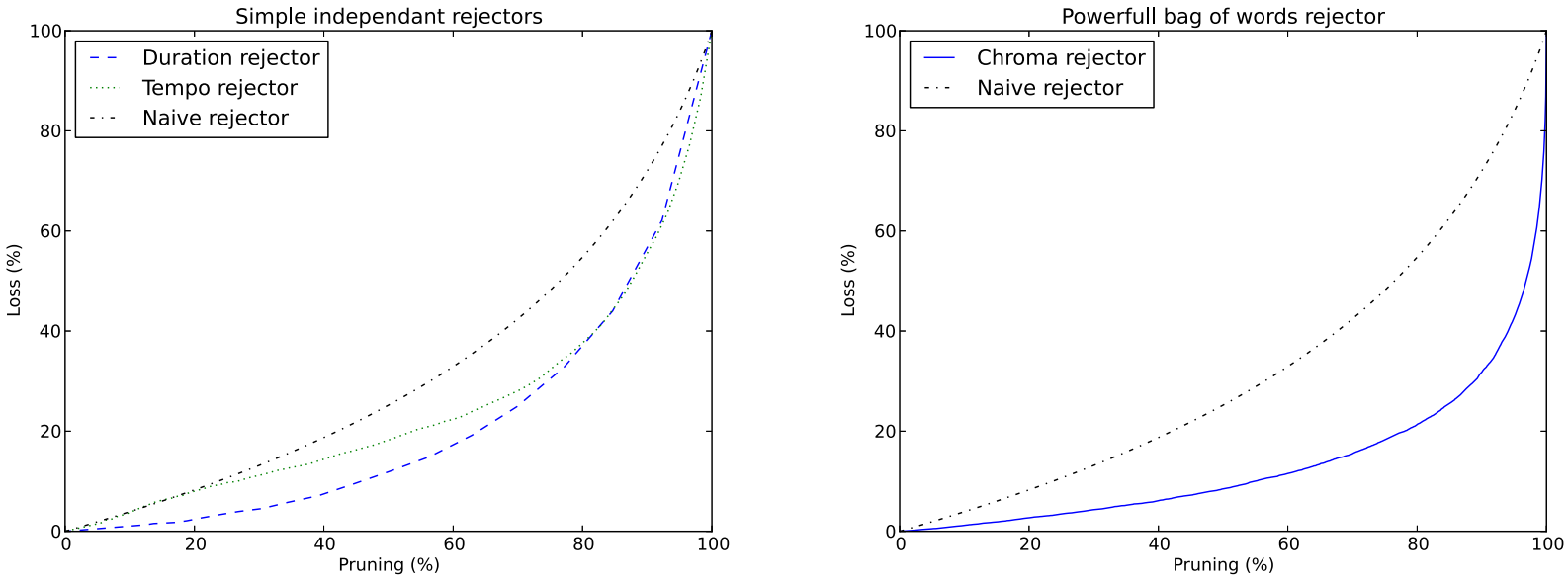
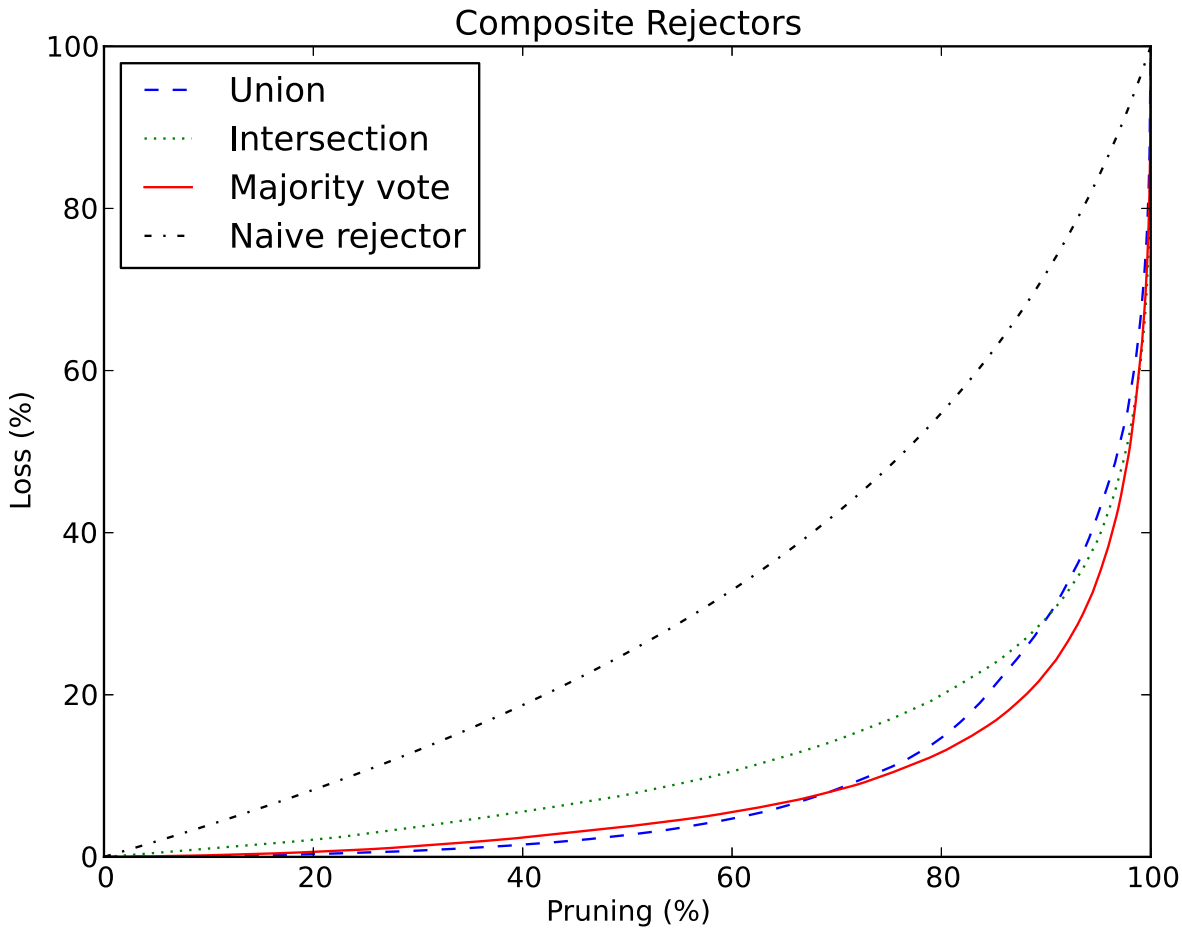



FIGURE: Simple and more complex rejectors

# Boolean combination of rejectors

Analysis of the behavior of combined rejectors using union, intersection and majority vote combinations.



# Combination of weak low-level rejectors

Investigation of the combination of weak rejectors using probabilistic and boolean fusion schemes.

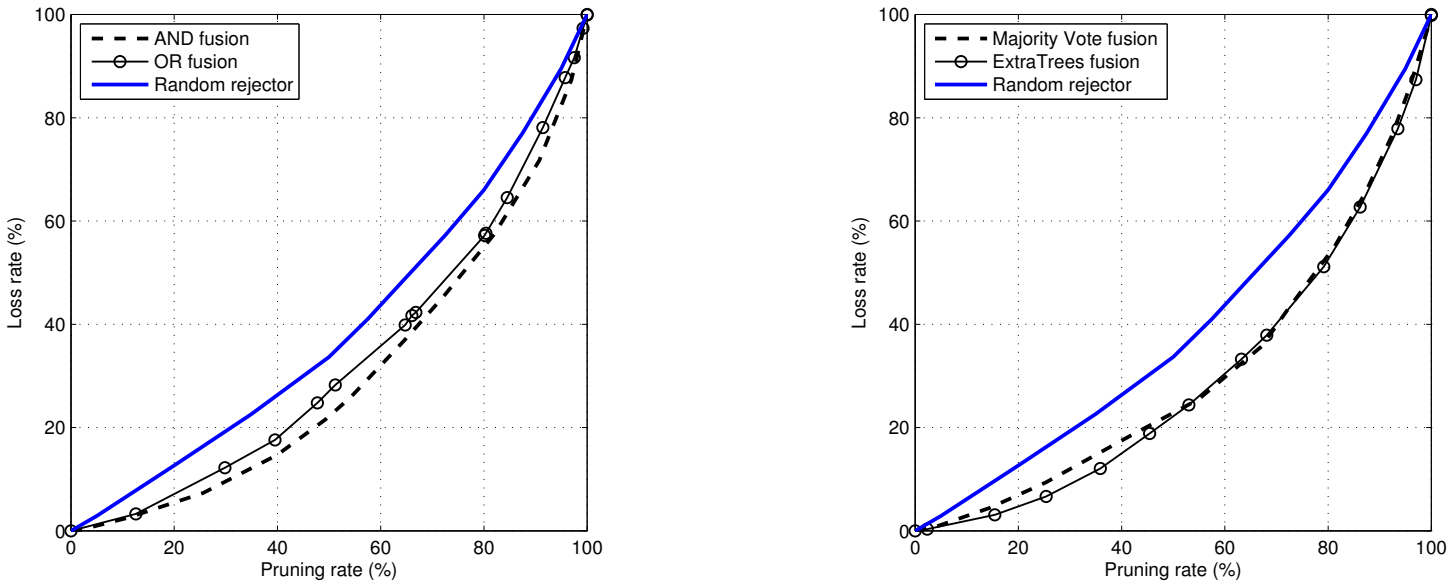
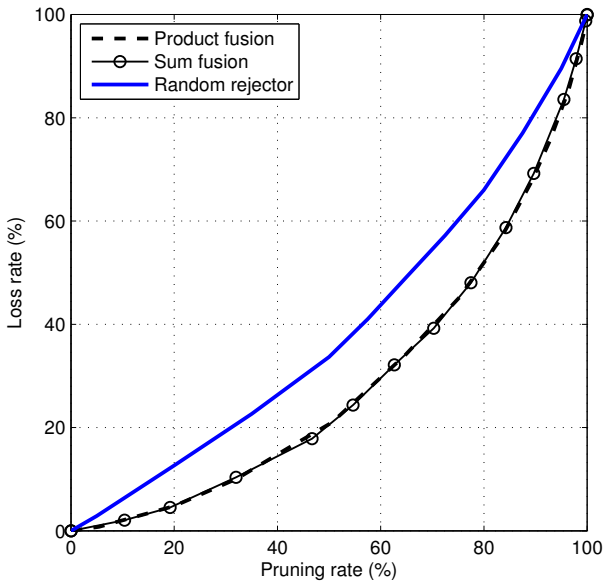



FIGURE: Boolean and meachine learning fusion of rejectors

# Combination of weak low-level rejectors

Investigation of the combination of weak rejectors using probabilistic and boolean fusion schemes.



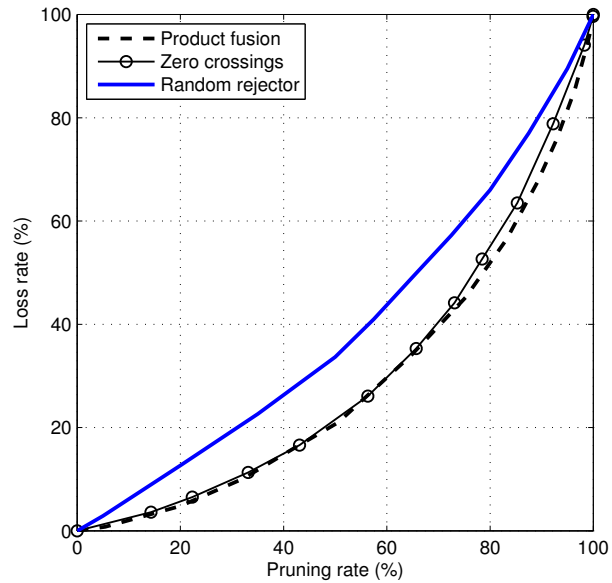




FIGURE: Probabilistic fusion of classifiers

# Contents

# Conclusions

Summary of the accomplished work so far.

- Formulated the problem of cover songs identification and the organization of compatible databases.
- Developped a new evaluation space based on Pruning and Loss.
- Developped a fast evaluation algorithm compatible with the PL space and other spaces.
- Designed an evaluation framework software to experiment quickly and easily with cover songs recognition problems.
- Experienced features with the framework.

# Future Work

- Improve the evaluation framework.
- Experiment with many features in the MSD.
- Explore new combinations of rejectors : e.g. machine learning, IBCAll, etc.
- Experiment with real audio data and integrate it to the framework.

Université de Liège

Queen Mary University of London

# Thank you for your attention !

Looking forward to work with you !

# Contents

# References I


[1] BERTIN-MAHIEUX, T. *Large-Scale Pattern Discovery in Music*. PhD thesis, Columbia University, 2013.

[2] CASEY, M., AND SLANEY, M. Fast recognition of remixed audio. In *Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)* (2007).

[3] ELLIS, D., AND POLINER, G. Identifying cover songs with chroma features and dynamic programming beat tracking. In *Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)* (2007), vol. 4.

[4] GROSCHE, P., AND MULLER, M. Toward characteristic audio shingles for efficient cross-version music retrieval. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on* (2012), IEEE, pp. 473–476.

[5] KIM, S., UNAL, E., AND NARAYANAN, S. Fingerprint extraction for classical music cover song identification. In *IEEE Int. Conf. Multimedia and Expo (ICME)* (2008), pp. 1261–1264.

# References II


[6] KURTH, F., AND MULLER, M. Efficient index-based audio matching. *Audio, Speech, and Language Processing, IEEE Transactions on 16*, 2 (2008), 382–395.

[7] LU, Y., AND CABRERA, J. Large scale similar song retrieval using beat-aligned chroma patch codebook with location verification. In *SIGMAP* (2012), pp. 208–214.

[8] OSMALSKYJ, J., PIÉRARD, S., VAN DROOGENBROECK, M., AND EMBRECHTS, J.-J. Efficient database pruning for large-scale cover song recognition. In *Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)* (Vancouver, Canada, May 2013), pp. 714–718.

[9] OSMALSKYJ, J., VAN DROOGENBROECK, M., AND EMBRECHTS, J.-J. Performances of low-level audio classifiers for large-scale music similarity. In *International Conference on Systems, Signals and Image Processing (IWSSIP)* (Dubrovnik, Croatia, May 2014), pp. 91–94.

[10] VAN BALEN, J., BOUNTOURIDIS, D., WIERING, F., AND VELTKAMP, R. Cognition-inspired descriptors for scalable cover song retrieval. In *Int. Symp. Music Inform. Retrieval (ISMIR)* (Taipei, Taiwan, Oct. 2014), pp. 379–384.