Music Information Retrieval 2.0

Markus Schedl
Peter Knees

{markus.schedl, peter.knees}@jku.at

Department of Computational Perception
Johannes Kepler University (JKU)
Linz, Austria
Why 2.0?

Music Information Retrieval (MIR) from a very traditional IR perspective is to have a short piece of music (symbolic, audio) and to find relevant pieces in a repository (e.g., that exact piece, “similar sounding” pieces, cover versions)

MIR spans more than that:

• MIR is not only query-by-example retrieval
• MIR is about music retrieval in general, i.e., also (textual) metadata and not just musical representations alone
• Moreover, “MIR 2.0” deals with vast amounts of data + utilizes the power-of-the-crowds, e.g., for “semantic” song annotations
Who are we?

**Markus Schedl**
Assistant Professor at the  
**Department of Computational Perception / Johannes Kepler University Linz**
M.Sc. in Computer Science from Vienna University of Technology  
Ph.D. in Computational Perception from Johannes Kepler University Linz  
M.Sc. in Int'l Business Administration from Vienna University of Economics and Business Administration

*Research interests:* social media mining, music and multimedia information retrieval, recommender systems, information visualization, and intelligent/personalized user interfaces

**Peter Knees**
Assistant Professor at the  
**Department of Computational Perception / Johannes Kepler University Linz**
M.Sc. in Computer Science from Vienna University of Technology  
Ph.D. in Computer Science from Johannes Kepler University Linz

*Research interests:* music and web information retrieval, multimedia, user interfaces, recommender systems, digital media arts
Overview

Introduction to Music Information Retrieval

Content-based Feature Extraction

Context- and Web-based Methods

Future Directions
Introduction to MIR

1. Definitions
2. Applications
3. Typical Tasks and Challenges
4. Types of Computational Features
Introduction to MIR: Definitions

“MIR is a multidisciplinary research endeavor that strives to develop innovative content-based searching schemes, novel interfaces, and evolving networked delivery mechanisms in an effort to make the world’s vast store of music accessible to all.”

[Downie, 2004]

“...actions, methods and procedures for recovering stored data to provide information on music.”

[Fingerhut, 2004]

“MIR is concerned with the extraction, analysis, and usage of information about any kind of music entity (for example, a song or a music artist) on any representation level (for example, audio signal, symbolic MIDI representation of a piece of music, or name of a music artist).

[Schedl, 2008]
Applications

Automatic Playlist Generation

Pandora.com

- continuously plays similar music
- based on the *Music Genome Project*
- manually annotated tracks
- songs can be rated
Applications

Browsing Music Collections

- music collections become larger and larger (on PCs as well as on mobile players)

- most UIs of music players still only allow organization and searching by textual properties according to scheme \((\text{genre-})\text{artist-}\text{album-}\text{track}\)

→ novel and innovative strategies to access music are sought in MIR

"intelligent iPod“ by CP@JKU
[Schnitzer et al., MUM 2007]
Browsing Music Collections

IFS @
TU Wien, 2006

[Mayer et al.,
ISMIR 2006]
Applications

“Musicream“,  
[Goto and Goto, ISMIR 2005]
Applications

Audio Fingerprinting

Query-by-example:
excerpt of a song (potentially recorded in low quality) used to identify the piece

Query-by-humming:
input is not excerpt of a song, but melody hum by the user

www.shazam.com
www.musicline.de/de/melodiesuche
www.soundhound.com
Applications

„Yanno“ – Chord Detection in Youtube videos, C4DM@QMUL, 2012
Applications

Real-time Music Following
Music Information Retrieval 2.0

Markus Schedl, Peter Knees
{markus.schedl, peter.knees}@jku.at | http://www.cp.jku.at

Applications

Geospatial Popularity Estimation
Applications

Geospatial Listening Patterns

[Schedl and Hauger, AdMIRe 2012]
Applications

Auto-tagging / Retrieval by Tag

[Sordo, PhD 2012]
Typical Tasks and Challenges

- feature extraction (audio-based vs. context-based approaches)
- similarity measurement, recommendation, automated playlist generation (last.fm, Pandora, Echo Nest, ...)
- user interfaces, visualization, and interaction
- audio fingerprinting (copyright infringement detection, music identification services like shazam.com or musicbrainz.org)
- voice and instrument recognition, speech/music discrimination
- structural analysis, alignment, and transcription (segmentation, self-similarities, music summarization, audio synthesis, audio and lyrics alignment, audio to score alignment aka score following, and audio to score transcription)
- optical music recognition (OMR)
- classification and evaluation (ground truth definitions, quality measurement, e.g. for feature extraction algorithms, genre classification)
The Feature Extraction Triangle

**music content**
- rhythm patterns
- MFCC models
- melodiousness
- percussiveness
- loudness

**music context**
- collaborative tags
- song lyrics
- album cover artwork
- artist's background
- playlist co-occurrences

**musical perception**

**user context**
- activities
- social context
- spatio-temporal context
- physiological aspects

[Schidl and Knees, *Personalization in Multimodal Music Retrieval*, Proc. AMR 2011]
Music Content vs. Music Context

Advantages of Content Analysis

• Features can be extracted from any audio file
• No other data or community necessary
• No cultural biases (i.e., no popularity bias, no subjective ratings etc.)

Advantages of Context Analysis

• Capture aspects beyond pure audio signal
• No audio file necessary
• Usually, user-based features are closer to what users want
Music Content vs. Music Context

Challenges for Context-based Feature Extractors

- Dependence on availability of sources (Web pages, tags, playlists, ...)
- Popularity of artists may distort results
- Cold start problem of community-based systems
  (newly added entities do not have any information associated, e.g. user tags,
   users’ playing behavior)
- Hacking and vandalism (cf. last.fm tag “brutal death metal”)
- Bias towards specific user/listener groups
  (e.g., young, Internet-prone, metal listeners in last.fm)
- (Reliable) data often only available on artist level
- Artist names that equal common speech terms

Challenge for both Content and Context Analysis

- Extraction of relevant features from noisy signal
Content-based Feature Extraction

1. Categorization of Content-based Features
2. Scheme of a Content-based Feature Extractor
3. Different Feature Extractors (low-level, mid-level, high-level)
Categorization of Content-based Features

Level of abstraction:

**low-level**
- closest to audio signal (e.g., energy, zero-crossing-rate);
  - app: audio identification

**mid-level**
- aggregate low-level, apply psycho-acoustic models (e.g., MFCC, FP);
  - app: similarity estimation, query-by-humming

**high-level**
- directly meaningful to end user (e.g., semantic tags learned from audio features);
  - app: autotagging, genre classification
Categorization of Content-based Features

Temporal scope:

*instantaneous*
feature is valid for a “point in time”, time resolution of ear is several msec!

*segment*
feature is valid for a segment, e.g., phrase, chorus (on a high level), or a chunk of X consecutive seconds in the audio signal

*global*
feature is valid for whole audio excerpt or piece of music
Categorization of Content-based Features

Acoustic property to describe:

- timbre
- rhythm
- pitch and melody
- chords and harmony
Categorization of Content-based Features

Domain:

*Time domain*
consider signal in *time/amplitude* representation

*Frequency domain*
consider signal in *frequency/magnitude* representation, Fast Fourier Transform
Scheme of a Content-based Feature Extractor

- Analog signal
- Pulse Code Modulation (PCM)
- Framing
- Windowing
- Time domain feature calculation
- Frequency domain feature calculation
- Aggregation, model building (mean, median, sum, GMM, HMM)
- Feature value, vector, or matrix
Analog-Digital-Conversion (ADC)

Problems that may occur in ADC:

**quantization error**: difference between the actual analog value and quantized digital value

**solution**: finer resolution (use more bits for encoding), common choice in music encoding: 16 bits per channel

Due to **Nyquist–Shannon Sampling Theorem**, frequencies above ½ of sampling frequency (Nyquist frequency) are discarded or heavily distorted

**solution**: choose a sampling frequency that is high enough (e.g. 44,100 Hz for Audio CDs)

PCM: analog signal is sampled at equidistant intervals and quantized in order to store it in digital form (here with 4 bits)
Frame
In short-time signal processing, pieces of music are cut into segments of fixed length, called frames, which are processed one at a time; typically, a frame comprises 256 - 4096 samples.

Windowing
To calculate the FFT, we need a periodic signal; thus, the PCM magnitude values of each frame are multiplied point-by-point with a suited function

e.g. Hanning window (Julius von Hann, Austrian meteorologist)

\[ h(x) = \frac{1}{2} \left[ 1 + \cos\left( \frac{\pi x}{n} \right) \right] \]

\( n \)...number of samples in frame
Fourier Transform
(after “Jean Baptiste Joseph Fourier”)

• transformation of the signal from **time domain** (time vs. amplitude) to **frequency domain** (frequency vs. magnitude)

• theorem: any continuous periodic function with a period of $2\pi$ can be represented as the sum of sine and/or cosine waves

• implication: any (analog) audio signal can be decomposed into an infinite number of overlapping waves (of different frequencies) given that it is periodic (windowing!)

• if the domain of the input function (audio signal) is discrete as in our case: → **Discrete Fourier Transform (DFT)**

• in practice: DFT is efficiently calculated via **Fast Fourier Transform (FFT)** by [Cooley and Tukey, 1965]
Critical Bands (Bark Scale)

- perceptually uniform measure of frequency
- reflects characteristics of human auditory system (our auditory perception is logarithmic!)
- measured according to bark scale
- used for psychoacoustic postprocessing
Representation as Short-time Fourier Transform (STFT)
Low-Level Feature: Root-Mean-Square (RMS) Energy
(aka RMS power, RMS level, RMS amplitude)

scope: time domain

calculation: \[ RMS_t = \sqrt{\frac{1}{K} \sum_{k=t\cdot K}^{(t+1)\cdot K-1} s(k)^2} \]

remarks:
+ beat-related feature, can be used for beat detection
+ related to perceived intensity
+ good loudness estimation
– discriminative power not clear

\( M_t(n) \)...magnitude in frequency domain at frame \( t \) and frequency bin \( n \)
\( s(k) \)...amplitude of \( k^{th} \) sample in time domain
\( K \)...frame size (number of samples in each frame)
\( N \)...number of highest frequency band
RMS Energy:
Illustration
Mid-Level Feature Extractor: Fluctuation Patterns

Idea:
measure how strong and fast beats are played within certain perceptually adjusted frequency bands

[Pampalk et al., ACM Multimedia 2002]
Fluctuation Patterns: Illustration

Floorilla, Anthem #5

Nightwish, Come Cover Me

Clannad, Caislean Oir

Mozart, Piano Sonata in C Major, K.330
High-Level Feature Extractor:
Block-Level Framework (BLF)  

Most low-level and mid-level feature extractors apply the Bag-of-Frames (BoF) model, i.e., features calculated for each frame are treated in an unordered fashion when aggregating.

→ Loss of Temporal Information

However, for most non-trivial music relations, temporal information is vital.

Idea of BLF:
Process blocks of frames instead of single frames.
A Typical Frame-Level Similarity Algorithm

1) digitized audio signal (PCM)

2) transform to frequency domain (FFT)

3) extract frame-level features

4) model each song as distribution of its feature vectors → **Bag of Frames (BoF)**
A Typical **Frame-Level Similarity Algorithm**: *How to measure similarity?*

As distance function the **KL-Divergence** (Kullback–Leibler), a measure of the relative entropy of two probability distributions, is frequently used.

\[
D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} \, dx
\]
High-Level Feature Extractor: Block-Level Framework (BLF)

Idea:
process audio signal in blocks instead of single frames

- better capability to capture temporal information
- entire feature extraction process is defined on block-level
High-Level Feature Extractor: Block-Level Framework (BLF)

Aggregating Block-Level to Song-Level Features

To obtain one feature vector for the entire song, a generalization step is performed, which is realized by an aggregation function.
High-Level Feature Extractor: Block-Level Framework (BLF)

BLF differ significantly from other features.

- **Advantages:**
  - Vector Space Model: The whole mathematical toolbox of vector spaces is available.
  - easy to use in classification
  - song models can be visualized

- **Disadvantages:**
  - high dimensional feature space
High-Level Feature Extractor: Block-Level Framework (BLF)

Several Block-Level Features have been defined in the Block-Processing Framework:

1. Spectral Pattern (SP)
2. Delta-Spectral Pattern (DSP)
3. Variance Delta-Spectral Pattern (VDSP)
4. Logarithmic Fluctuation Pattern (LFP)
5. Correlation Pattern (CP)
6. Spectral Contrast Pattern (SCP)

Aggregating Block-Level to Song-Level Features
High-Level Feature Extractor: Block-Level Framework (BLF)

Spectral Pattern (SP)

- An audio descriptor that captures the frequency content of an audio file
- Each frequency band of an audio block is simply sorted along the time axis.

\[
\text{SP} = \begin{bmatrix}
\text{sort}(b_{1,1} & \cdots & b_{1,W}) \\
\vdots & \ddots & \vdots \\
\text{sort}(b_{H,1} & \cdots & b_{H,W})
\end{bmatrix}
\]

- The 0.9-percentile is used as aggregation function.
High-Level Feature Extractor: Block-Level Framework (BLF)

Delta-Spectral Pattern (DSP)

- First the delta spectrum, i.e., the difference of the original and a delayed version (by 3 frames) is computed to emphasize onsets.
- The same strategy as for the SP is followed – all frequency bands are sorted.
- The 0.9-percentile is used as aggregation function.
High-Level Feature Extractor: Block-Level Framework (BLF)

Variance Delta-Spectral Pattern (VDSP)

- Similar to the DSP
- The variance is used instead of the 0.9-percentile as aggregation function.
- Captures temporal variations of the onset strength
High-Level Feature Extractor: Block-Level Framework (BLF)

Logarithmic Fluctuation Pattern (LFP)

- Modified version of the Fluctuation Patterns
- The Cent scale is used instead of the Sone representation.
- The periodicity dimension of the pattern is logarithmically scaled to yield a more tempo invariant representation.
High-Level Feature Extractor: Block-Level Framework (BLF)

Correlation Pattern (CP)

- Reduce the Cent spectrum to 52 frequency bands
- Captures the temporal relation of the frequency bands
- Compute the pairwise linear correlation between each frequency band.
  \[ r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{(n - 1)s_xs_y} \]
- The 0.5-percentile is used as aggregation function.

ECIR 2012, Barcelona, Spain
High-Level Feature Extractor: Block-Level Framework (BLF)

Spectral Contrast Pattern (SCP)

- Compute the **spectral contrast** per frame to estimate the "tone-ness".
- This is performed separately for 20 frequency bands of the Cent spectrum.
- Sort the spectral contrast values of each frequency band along the whole block.
- The aggregation function is the 0.1-percentile.
High-Level Feature Extractor:
Block-Level Framework (BLF)

Spectral Contrast Pattern (SCP)

- Compute the **spectral contrast** per frame to estimate the "tone-ness".
- This is performed separately for 20 frequency bands of the Cent spectrum.
- Sort the spectral contrast values of each frequency band along the whole block.
- The aggregation function is the 0.1-percentile.
High-Level Feature Extractor: Block-Level Framework (BLF)

Spectral Contrast Pattern (SCP)

- Compute the spectral contrast per frame to estimate the "tone-ness".
- This is performed separately for 20 frequency bands of the Cent spectrum.
- Sort the spectral contrast values of each frequency band along the whole block.
- The aggregation function is the 0.1-percentile.
High-Level Feature Extractor: Block-Level Framework (BLF)

Spectral Contrast Pattern (SCP)

- Compute the **spectral contrast** per frame to estimate the \``tone-ness\``.

- This is performed separately for 20 frequency bands of the Cent spectrum.

- Sort the spectral contrast values of each frequency band along the whole block.

- The aggregation function is the 0.1-percentile.
High-Level Feature Extractor: Block-Level Framework (BLF)

Spectral Contrast Pattern (SCP)

- Compute the **spectral contrast** per frame to estimate the "tone-ness".
- This is performed separately for 20 frequency bands of the Cent spectrum.
- Sort the spectral contrast values of each frequency band along the whole block.
- The aggregation function is the 0.1-percentile.
High-Level Feature Extractor: Block-Level Framework (BLF)

Spectral Contrast Pattern (SCP)

- Compute the spectral contrast per frame to estimate the "tone-ness".

- This is performed separately for 20 frequency bands of the Cent spectrum.

- Sort the spectral contrast values of each frequency band along the whole block.

- The aggregation function is the 0.1-percentile.
High-Level Feature Extractor: Block-Level Framework (BLF)

How to estimate song similarities for multiple block-level features?

\[ D(S_1, S_2) = ? \]

Two music similarity algorithms possible:

- The first one is based on directly comparing and combining the block-level features (BLS).

- The second similarity measure performs a mapping over a semantic tag space (TAG).
High-Level Feature Extractor: Block-Level Framework (BLF)

- **Estimate** song similarities separately for each pattern (by computing Manhattan distance).

- **Combine** the similarity estimates of the individual patterns into a single result.
High-Level Feature Extractor: Block-Level Framework (BLF)

Naïve approach: linearly weighted combination of BLFs

Problem: Similarity estimates of the different patterns (block-level features) have different scales.

→ special normalization strategy is used: **Distance Space Normalization**
High-Level Feature Extractor: Block-Level Framework (BLF)

- Operates on the distance matrix
- Each distance $D_{n,m}$ is normalized using Gaussian normalization.
- Mean and standard deviation are computed over both column and row of the distance matrix.
- Each distance has its own normalization parameters.

**Observation:** The operation itself can improve the nearest neighbor classification accuracy.
High-Level Feature Extractor:
Block-Level Framework (BLF)

TAG: The Semantic Space Approach

Tag Model Comparison
Calculate Similarity in Semantic Space (Manhattan distance)
nepTune

Browsing Music Collections...
...can be fun
nepTune
Clustering of music pieces

Each song corresponds to point in feature (similarity) space

Self-organizing Map

High-dimensional data (content-based features) is projected to 2-dim. plane

Number of pieces per cluster → landscape height profile
nepTune

Automatic description of landscape via Web term extraction

artist names (ID3)

Music dictionary

Term goodness

\[ G_{w,j} = \frac{F_{w,j}^2}{\sum_i F_{w,i}} \]
nepTune: Demo Video
Music Information Retrieval 2.0

Part II – Context-based MIR

Markus Schedl
Peter Knees

Department of Computational Perception
Johannes Kepler University (JKU)
Linz, Austria
Context- and Web-based Methods

“Music is more than just the signal”

“Similarity” as perceived by humans is not exclusively determined by the audio but also by the outside world, i.e., the context

Influenced by, e.g., marketing strategies, political views of artists, language of songs, decade of activity, etc.
(also listening context, mood, peers – see later)

Analysis of audio alone can’t capture these aspects
Context- and Web-based Methods

In the following, contextual data refers to extended meta-data, usually

- Generated by users
- Unstructured data-sources
- Accessible via the Web

Two main classes of approaches covered in the following

- Text processing
- Co-occurrence analysis

As for content-based methods, similarity is a central concept
Text-based Approaches

Data sources:

- Web pages retrieved via Web search engines
- microblogs on twitter
- product reviews
- semantic tags
- lyrics
Text-based Similarity and Genre Classification

Use Web data to transform the **music similarity** task into a **text similarity** task.

Allows to use the full armory of IR methods, typically…

- Bag-of-words, Vector Space Model
- Stopword removal, dictionaries, term selection
- $\text{TF} \cdot \text{IDF}$
- Latent Semantic Indexing
- Part-of-Speech tagging
- Named Entity Detection
- Sentiment analysis

Large range of possible similarity measures

- Overlap, Manhattan, Euclidean, Cosine, etc.
Related Web Pages as Text Source
Related Web Pages as Text Source

Google™ → Web pages → features

similar to...?
Related Web Pages as Text Source

Using search engines and queries such as
“artist” +music
“artist” +music +review
[Whitman and Lawrence, ICMC 2002] [Baumann and Hummel, WEDELMUSIC 2003]
[Knees et al., ISMIR 2004]

Analyze result page directly or download up to top 100 Web pages
(combine into one “virtual document” or analyze separately)

Apply “IR magic”

Applicable for similarity estimation, classification, retrieval, annotation

(NB: Most discriminating terms between genres are artist names and album/track titles)
Large-Scale Study [Schedl et al., TOIS 2011]

Investigating different aspects in modeling artist term profiles from Web pages (9,200 experiments):

- **term frequency**

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Description</th>
<th>Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF_A</td>
<td>Formulation used for binary match SB = b</td>
<td>[ r_{d,t} = \begin{cases} 1 &amp; \text{if } t \in \mathcal{I}_d \ 0 &amp; \text{otherwise} \end{cases} ]</td>
</tr>
<tr>
<td>TF_B</td>
<td>Standard formulation SB = t</td>
<td>[ r_{d,t} = f_{d,t} ]</td>
</tr>
<tr>
<td>TF_C</td>
<td>Logarithmic formulation</td>
<td>[ r_{d,t} = 1 + \log_e f_{d,t} ]</td>
</tr>
<tr>
<td>TF_C2</td>
<td>Alternative logarithmic formulation suited for ( f_{d,t} &lt; 1 )</td>
<td>[ r_{d,t} = \log_e(1 + f_{d,t}) ]</td>
</tr>
<tr>
<td>TF_C3</td>
<td>Alternative logarithmic formulation as used in ltc variant</td>
<td>[ r_{d,t} = 1 + \log_2 f_{d,t} ]</td>
</tr>
<tr>
<td>TF_D</td>
<td>Normalized formulation</td>
<td>[ r_{d,t} = \frac{f_{d,t}}{\sum_{d'} f_{d',t}} ]</td>
</tr>
<tr>
<td>TF_E</td>
<td>Alternative normalized formulation. Similar to [55] we use ( K = 0.5 ). SB = n</td>
<td>[ r_{d,t} = K + (1 - K) \cdot \frac{f_{d,t}}{\sum_{d'} f_{d',t}} ]</td>
</tr>
<tr>
<td>TF_F</td>
<td>Okapi formulation, according to [55, 36]. For W we use the vector space formulation, i.e., the Euclidean length.</td>
<td>[ r_{d,t} = \frac{f_{d,t}}{f_{d,t} + W_d / \sum_{d \in D} W_d} ]</td>
</tr>
<tr>
<td>TF_G</td>
<td>Okapi BM25 formulation, according to [35].</td>
<td>[ r_{d,t} = \frac{(k_1 + 1) \cdot f_{d,t}}{f_{d,t} + k_1 \cdot \frac{(1-b) + b \cdot \frac{W_d}{\sum_{d \in D} W_d}}{W_d}} ] [ k_1 = 1.2, b = 0.75 ]</td>
</tr>
</tbody>
</table>
Investigating different aspects in modeling artist term profiles from Web pages (9,200 experiments):

- **term frequency**
- **inverse document frequency**

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Description</th>
<th>Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDF_A</td>
<td>Formulation used for binary match ( SB = x )</td>
<td>( w_t = 1 )</td>
</tr>
<tr>
<td>IDF_B</td>
<td>Logarithmic formulation ( SB = f )</td>
<td>( w_t = \log_a \left( 1 + \frac{N}{f_t} \right) )</td>
</tr>
<tr>
<td>IDF_B2</td>
<td>Logarithmic formulation used in ltc variant</td>
<td>( w_t = \log_a \left( \frac{N}{f_t} \right) )</td>
</tr>
<tr>
<td>IDF_C</td>
<td>Hyperbolic formulation</td>
<td>( w_t = \frac{1}{f_t} )</td>
</tr>
<tr>
<td>IDF_D</td>
<td>Normalized formulation</td>
<td>( w_t = \log_a \left( 1 + \frac{f_{tm}}{f_t} \right) )</td>
</tr>
<tr>
<td>IDF_E</td>
<td>Another normalized formulation ( SB = p )</td>
<td>( w_t = \log_a \left( \frac{N - f_t}{f_t} \right) )</td>
</tr>
<tr>
<td>IDF_F</td>
<td>Signal</td>
<td>( w_t = s_t )</td>
</tr>
<tr>
<td>IDF_G</td>
<td>Signal-to-Noise ratio</td>
<td>( w_t = \frac{s_t}{n_t} )</td>
</tr>
<tr>
<td>IDF_H</td>
<td>Entropy measure</td>
<td>( w_t = \left( \max_{t' \in T} n_{t'} \right) - n_t )</td>
</tr>
<tr>
<td>IDF_I</td>
<td>Okapi BM25 IDF formulation, according to [35, 31]</td>
<td>( w_t = \log_a \frac{N - f_t + 0.5}{f_t + 0.5} )</td>
</tr>
<tr>
<td>IDF_J</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Large-Scale Study [Schedl et al., TOIS 2011]

Investigating different aspects in modeling artist term profiles from Web pages (9,200 experiments):
- term frequency
- inverse document frequency
- virtual document modeling:
  *concatenate* all Web pages/posts of the artist or perform *aggregation* via mean, max, etc.
Large-Scale Study  

[Schedl et al., TOIS 2011]

Investigating different aspects in modeling artist term profiles from Web pages (9,200 experiments):

- **term frequency**
- **inverse document frequency**
- **virtual document modeling:**
  - *concatenate* all Web pages/posts of the artist or perform *aggregation* via mean, max, etc.
- **normalization with respect to document length**

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Description</th>
<th>Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORM_NO</td>
<td>No normalization.</td>
<td></td>
</tr>
<tr>
<td>NORM_SUM</td>
<td>Normalize sum of each virtual document’s term feature vector to 1.</td>
<td>$\sum_{t \in T_d} r_{d,t} = 1$</td>
</tr>
<tr>
<td>NORM_MAX</td>
<td>Normalize maximum of each virtual document’s term feature vector to 1.</td>
<td>$\max_{t \in T_d} r_{d,t} = 1$</td>
</tr>
</tbody>
</table>
Large-Scale Study

Investigating different aspects in modeling artist term profiles from Web pages (9,200 experiments):
- term frequency
- inverse document frequency
- virtual document model: 
  - concatenate all Web pages
- normalization with respect to document length
- similarity measure

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Description</th>
<th>Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM_INN</td>
<td>Inner Product</td>
<td>( S_{d_1,d_2} = \sum_{t \in T_{d_1,d_2}} (w_{d_1,t} \cdot w_{d_2,t}) )</td>
</tr>
<tr>
<td>SIM_COS</td>
<td>Cosine Measure</td>
<td>( S_{d_1,d_2} = \frac{\sum_{t \in T_{d_1,d_2}} (w_{d_1,t} \cdot w_{d_2,t})}{\sqrt{\sum_{t \in T_{d_1}} w_{d_1,t}^2 \cdot \sum_{t \in T_{d_2}} w_{d_2,t}^2}} )</td>
</tr>
<tr>
<td>SIM_DIC</td>
<td>Dice Formulation</td>
<td>( S_{d_1,d_2} = \frac{2 \sum_{t \in T_{d_1,d_2}} (w_{d_1,t} \cdot w_{d_2,t})}{w_{d_1}^2 + w_{d_2}^2} )</td>
</tr>
<tr>
<td>SIM_JAC</td>
<td>Jaccard Formulation</td>
<td>( S_{d_1,d_2} = \frac{\sum_{t \in T_{d_1,d_2}} (w_{d_1,t} \cdot w_{d_2,t})}{w_{d_1}^2 + w_{d_2}^2 - \sum_{t \in T_{d_1,d_2}} w_{d_1,t} \cdot w_{d_2,t}} )</td>
</tr>
<tr>
<td>SIM_OVL</td>
<td>Overlap Formulation</td>
<td>( S_{d_1,d_2} = \frac{\sum_{t \in T_{d_1,d_2}} (w_{d_1,t} \cdot w_{d_2,t})}{\min(w_{d_1}^2, w_{d_2}^2)} )</td>
</tr>
<tr>
<td>SIM_EUC</td>
<td>Euclidean Similarity</td>
<td>( D_{d_1,d_2} = \sqrt{\sum_{t \in T_{d_1,d_2}} (w_{d_1,t} - w_{d_2,t})^2} )</td>
</tr>
<tr>
<td>SIM_JEF</td>
<td>Jeffrey Divergence-based Similarity</td>
<td>( S_{d_1,d_2} = \left( \max_{d_1',d_2'} (D_{d_1',d_2'}) \right) - D_{d_1,d_2} )</td>
</tr>
</tbody>
</table>

\( D(F,G) = \sum_t \left( f_t \log \frac{f_t}{m_t} + g_t \log \frac{g_t}{m_t} \right) \)

\( m_t = \frac{f_t + g_t}{2} \)
Large-Scale Study

Investigating different aspects in modeling artist term profiles from Web pages (9,200 experiments):
- term frequency
- inverse document frequency
- virtual document modeling:
  - concatenate all Web pages/posts of the artist
  - perform aggregation via mean, max, etc.
- normalization with respect to document length
- similarity measure
- index term set

<table>
<thead>
<tr>
<th>Abbr. / Term Set</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS_A - all_terms</td>
<td>C224a, QS_A: 38,133</td>
<td>All terms (stemmed) that occur in the corpus of the retrieved Twitter posts.</td>
</tr>
<tr>
<td></td>
<td>C224a, QS_M: 19,133</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C3ka, QS_A: 1,489,459</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C3ka, QS_M: 437,014</td>
<td></td>
</tr>
<tr>
<td>TS_S - scowl_dict</td>
<td>698,812</td>
<td>All terms that occur in the entire SCOWL dictionary.</td>
</tr>
<tr>
<td>TS_N - artist_names</td>
<td>224 / 3,000</td>
<td>Names of the artists for which data was retrieved.</td>
</tr>
<tr>
<td>TS_D - dictionary</td>
<td>1,398</td>
<td>Manually created dictionary of musically relevant terms.</td>
</tr>
<tr>
<td>TS_L - last.fm_toptags</td>
<td>250</td>
<td>Overall top-ranked tags returned by last.fm’s Tags.getTopTags function.</td>
</tr>
<tr>
<td>TS_F - freebase</td>
<td>3,628</td>
<td>Music-related terms extracted from Freebase (genres, instruments, emotions).</td>
</tr>
</tbody>
</table>
Large-Scale Study [Schedl et al., TOIS 2011]

Investigating different aspects in modeling artist term profiles from Web pages (9,200 experiments):
- term frequency
- inverse document frequency
- virtual document modeling: concatenate all Web pages/posts of the artist or perform aggregation via mean, max, etc.
- normalization with respect to document length
- similarity measure
- index term set
- query scheme

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Query Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>QS_A</td>
<td>“artist name”</td>
</tr>
<tr>
<td>QS_M</td>
<td>“artist name”+music</td>
</tr>
</tbody>
</table>
Large-Scale Study  [Schedl et al., TOIS 2011]

Investigating different aspects in modeling artist term profiles from Web pages (9,200 experiments):
- term frequency
- inverse document frequency
- virtual document modeling:
  - concatenate all Web pages/posts of the artist or perform aggregation via mean, max, etc.
- normalization with respect to document length
- similarity measure
- index term set
- query scheme

implemented in our CoMIRVA framework available from http://www.cp.jku.at/comirva
Interesting Findings [Schedl et al., TOIS 2011]

- modeling artists as **virtual documents** is preferable

- using query scheme “**artist**” +**music** outperforms “**artist**”

- **normalization** does not yield a statistically significant difference

- standard **cosine similarity** measure does not yield the very best results, but the most stable ones (varying other parameters)

- **consistent results** among the (top-ranked) variants for two collections

- **minor change in one parameter** can have a huge impact on performance

- overall winners in terms of term weighting functions:
  - TF_C3.IDF_I
  - TF_C3.IDF_H
  - TF_C2.IDF_I

  → logarithmic formulations for TF and IDF
“MusicSun” [Pampalk and Goto, ISMIR'07]

- Interactive “Artist Recommender”
- Recommendation is influenced/directed by selecting relevant similarity dimensions
- Combines different similarity measures

3 types of similarity: audio, web-based, word
overall similarity = weighted average of ranks
Web-based Texts for Indexing and Retrieval

Use Web data to transform **music retrieval** into a **text retrieval** task

Find associated (or associable) texts and use them instead of music

Allows for diverse and semantic queries (e.g., “chilled music”, “great riffs”)

**Search Sounds** [Celma et al., ISMIR 2006]:
Crawl list of RSS feeds and use Weblog entries to index pieces

**Squiggle** [Celino et al., SAMT 2006]:
Combines meta-data databases (like MusicBrainz) for rich indexing

**Gedoodle** [Knees et al., SIGIR 2007]:
Query Google and combine Web pages to index pieces
Gedoodle

[Knees et al., SIGIR 2007]

For each track: join 100 Google results of

- “artist” music
- “artist” “album” music review
- “artist” “title” music review -lyrics

Combine all pages into one virtual document

Create normalized TFIDF vector for each track

Include audio similarity for vector modification and dimensionality reduction
Gedoodle (Example queries)
Music Information Retrieval 2.0

Markus Schedl, Peter Knees
{markus.schedl, peter.knees}@jku.at | http://www.cp.jku.at

Music Information Extraction from Web Pages

Web data is a rich source for all types of meta-data and semantic relations

Apply methods from NLP, IE, Named Entity Detection to extract data

- **Genres, Moods, Similarities** using Rule Patterns [Geleijnse and Korst, ISMIR‘06]
- **Band Members and Line-Up** using Rule Patterns [Schedl and Widmer, AMR‘07]
- **Band Members, Discography, Artist Detection** using rules in GATE [Krenmair, JKU 2010]
- **Band Members, Discography** using Supervised Learning (in GATE) [Knees and Schedl, WOMRAD 2011]
- **Album cover** detection and extraction [Schedl et al., ECIR 2006]
Microblogs as Text Sources
Microblogs as Text Sources

Retrieving tweets from *twitter.com* using hashtags such as 
#nowplaying 
#np 
#itunes [Schedl et al., ISMIR 2011]

Highly compressed, non-natural language due to 140 character restriction

Artist/track identification and matching often challenging

#nowplaying also used for movies
Interesting Findings on Large-Scale Evaluation

- modeling artists as virtual documents is preferable

- using query scheme "artist" outperforms "artist" +music (sparsity!)

- normalization does not improve results! → 140-character limit

- term set has an important influence: TS_A best, but at highest cost! TS_N good compromise

- simple inner product as similarity measure performed better than cosine → consistent with observation for normalization

- no clear picture for TF and IDF variants (just don't use binary match!)
  TF_C2 (log) and TF_E performed best, on average
  IDF_B2 (log), IDF_J (Okapi), IDF_E (log) performed best, on average
Product Reviews as Text Sources
Product Reviews as Text Sources

Exploiting sources such as _amazon.com_ or _epinions.com_ [Hu et al., 2005]

- Allows for sentiment analysis and associated rating prediction
- Very prone to attacks (remedy: consider “helpfulness” ratings)
Community Tags as Text Sources
Tag Sources

- Community
e.g., last.fm

- Soundcloud (annotations along timeline)

- Games with a purpose
e.g., Tag-a-Tune
  [Law and von Ahn, CHI’09]

- Autotags (see before)
Community Tags as Text Sources

Treating collections of tags (e.g., from last.fm) as documents

[Pohle et al., ISMIR 2007] [Levy and Sandler, JNMR 2008] [Hu et al., ISMIR 2009]

• Retrieve tags for artist or track from last.fm
• Cleaning of noisy and redundant tags
  - manually
  - automatically [Geleijnse et al., ISMIR 2007]
• List of collected terms is treated as text document and TF·IDF’d
  [Levy & Sandler, ISMIR 2007]
• Optionally, LSA to reduce dimensionality
• Comparison of vectors via cosine similarity (or overlap score)

• Data often available in standardized fashion, dedicated terms for music
• Lower dimensionality
  (13,500 tags vs. >200,000 Web terms [Levy & Sandler, ISMIR 2007])
• Depends on community, needs annotators
• Hacking and Attacks!
Music Information Retrieval 2.0

Markus Schedl, Peter Knees

{markus.schedl, peter.knees}@jku.at | http://www.cp.jku.at

Lyrics as Text Source

Before day break there was none
And as it broke there was one
The Moon, the sun, it goes on 'n' on
The winter battle was won
The summer children were born
And so the story goes on 'n' on
Come woman if your life beats
Those we buried with the house keys
Smoke and feather where the fields are green
From here to eternity
Come woman on your own time

Around the world, around the world
Around the world, around the world
Around the world, around the world
Around the world, around the world
Around the world, around the world
Around the world, around the world
Around the world, around the world
Around the world, around the world
Around the world, around the world

ECIR 2012, Barcelona, Spain
Lyrics as Text Source

Topic Features

[Logan et al., ICME 2004]
- Typical topics for lyrics are distilled from a large corpus using (P)LSA ("Hate", "Love", "Blue", "Gangsta", "Spanish")
- Lyrics are transformed to topic-based vectors, similarity is calculated via $L_1$ distance
- Alternative approaches use TF-IDF with optional LSA and Stemming for Mood Categorization
  [Laurier et al., ISMIR 2008], [Hu et al., ISMIR 2009]

Rhyme Features [Mayer et al., ACM MM 2008] [Hirjee and Brown, ISMIR 2009]
- Phonetic transcription is searched for patterns of rhyming lines (AA, ABAB, AABB)
- Frequency of patterns + statistics like words per minute, punctuation freq. etc.

Other Features [Mahedero et al., ACM MM 2005] [Hirjee and Brown, ISMIR 2009]
- Language, structure
### Text-based Similarity Approaches: Summary

<table>
<thead>
<tr>
<th>Source</th>
<th>Web-Terms</th>
<th>Microblogs</th>
<th>Reviews</th>
<th>Tags</th>
<th>Lyrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Web pages</td>
<td>platform</td>
<td>shops, platform</td>
<td>Web service</td>
<td>portal</td>
</tr>
<tr>
<td>Community-based</td>
<td>depends</td>
<td>depends</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Level</td>
<td>artists</td>
<td>artists (tracks)</td>
<td>albums</td>
<td>artists (tracks)</td>
<td>tracks (artists)</td>
</tr>
<tr>
<td>Feature Dimensionality</td>
<td>very high</td>
<td>high</td>
<td>possibly high</td>
<td>moderate</td>
<td>possibly high</td>
</tr>
<tr>
<td>Specific Bias</td>
<td>low</td>
<td>low</td>
<td>personal</td>
<td>community</td>
<td>none</td>
</tr>
<tr>
<td>Potential Noise</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>moderate</td>
<td>low</td>
</tr>
</tbody>
</table>
Similarity from Co-Occurrences

Idea: expect entities that occur frequently in the same context to be similar

Data sources:

- page count estimates from Web search engines
- shared folders/search queries on the Gnutella file sharing network
- collaborative filtering on playcounts from last.fm
- occurrences in playlists
Search Engine Page Count Estimates

[Schedl et al., CBMI 2005]

For all pairs of artists: query “artist 1” “artist 2” +music +review
For each artist: query “artist” +music +review

Use page counts for sim. (results in quadratic page count matrix)

\[ \text{sim}_{pc-cp}(A_i, A_j) = \frac{1}{2} \cdot \left( \frac{pc(A_i, A_j)}{pc(A_i)} + \frac{pc(A_i, A_j)}{pc(A_j)} \right) \]

To avoid quadratic number of queries: download top 100 pages for each artist and parse for occurrences of other artists (linear compl.)

NB: asymmetry of pc matrix can be used to identify prototypical artists!
Shared Folders in a P2P Network

Make use of meta-data transmitted as files names or ID3 tags in P2P network OpenNap [Whitman and Lawrence, ICMC 2002] [Ellis et al., ISMIR 2002]

Information gathered from users' shared folders (no file downloads!)

Similarities via artist co-occurrences in collections (cond. prob.)

Sparse co-occurrence matrix

Experiments on Gnutella network [Shavitt and Weinsberg, AdMIRe 2009]:

• meta-data highly inconsistent

• can be used as song-based similarity measure and to estimate localized popularity/trends (matching IP addresses difficult!)
Last.fm Playcounts

Use *explicit* or *implicit* ratings of users or interpret number of plays of a song as a “rating”

Results in a user-track rating matrix

Use standard **collaborative filtering** approaches to predict similarities (or to recommend unknown music) e.g., [Resnick et al., CSCW’94]

Item-based: compare tracks by calculating similarity on vectors over all users

User-based: find similar users by comparing listening pattern vectors; use to find relevant/similar and yet unknown tracks
Playlist Co-Occurrences

Analysis of co-occurrences of artists and songs on radio station playlists (French radio station Fip) and compilation CD databases (CDDB) [Pachet et al., WEDELMUSIC 2001]

\[
s_{\text{mpl-cooc}}(A_i, A_j) = \frac{1}{2} \cdot \left[ \frac{cooc(A_i, A_j)}{cooc(A_i, A_i)} + \frac{cooc(A_j, A_i)}{cooc(A_j, A_j)} \right]
\]

Analysis of 29,000 playlists from “Art of the Mix” [Cano and Koppenberger, ISMIR'04]: artists similar if they co-occur in playlist (highly sparse)

Analysis of > 1 million playlists from “MusicStrands” [Baccigalupo et al., ISMIR 2008]:

- distance in playlists taken into account \( \beta_0 = 1, \beta_1 = 0.8, \beta_2 = 0.64 \)

\[
d_{\text{dist-pl-d}}(A_i, A_j) = \sum_{h=0}^{2} \beta_h \cdot [d_h(A_i, A_j) + d_h(A_j, A_i)]
\]

- playlist prediction using case-based reasoning
Co-occurrence-based Approaches: Summary

<table>
<thead>
<tr>
<th>Source</th>
<th>Web Co-Ocs</th>
<th>Playcounts</th>
<th>P2P nets</th>
<th>Playlists</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>search engines,</td>
<td>listening</td>
<td>shared</td>
<td>radio, compilations,</td>
</tr>
<tr>
<td></td>
<td>Web pages</td>
<td>service</td>
<td>folders</td>
<td>Web services</td>
</tr>
<tr>
<td>Community-based</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>depends on source</td>
</tr>
<tr>
<td>Level</td>
<td>artists</td>
<td>tracks</td>
<td>artists (tracks)</td>
<td>artists (tracks)</td>
</tr>
<tr>
<td>Specific Bias</td>
<td>&quot;wikipedia&quot;-bias</td>
<td>popularity</td>
<td>community</td>
<td>low</td>
</tr>
<tr>
<td>Potential Noise</td>
<td>high</td>
<td>low</td>
<td>high</td>
<td>low</td>
</tr>
</tbody>
</table>
Current and Future Challenges

Improving content- and context-based methods (scale and speed)

Most urgent challenge: Modelling **User Context**

**Perception of music is highly subjective**

Influenced by personal experiences, preferences, taste, etc.

Analysis of audio and Web data alone can’t capture these aspects

Essential for truly personalized recommendations
  (in contrast to e.g., collaborative filtering)
Dimensions of User Context

adapted from [Göker and Myrhaug, 2002]

**Environmental Context:** people & things surrounding user, temperature, noise, light, humidity

**Personal Context:**
- *physiological context*: weight, blood pressure, pulse, …
- *mental context*: stress level, mood, expertise, …

**Task Context:** current activities, running apps
Dimensions of User Context

adapted from [Göker and Myrhaug, 2002]

Social Context: relatives, friends, enemies, collaborators

Spatio-temporal Context: location, place, direction, speed, and time
User-Aware Music Recommendation on Android Phones

"Mobile Music Genius": music player for the Android platform

- collecting user context data while playing (next slides)
- adaptive system that learns user taste/preferences from implicit feedback (player interaction)

- two basic research questions:
  + Can we solve a binary classification problem (like/dislike) from context features and feedback? (assuming skipping indicates <dislike>, actively selecting a song indicates <like>)
  + Can we relate user context features to music the user wants to listen to?

- ultimate aim: dynamically and seamlessly update the user's playlist according to his/her current context
User-Aware Music Recommendation on Android Phones

(ongoing Master's Thesis G. Breitschopf)
User Context Features from Android Phones

Time: timestamp, time zone

Personal: userID/eMail, gender, birthdate

Device: deviceID (IMEI), sw version, manufacturer, model, phone state, connectivity, storage, battery, various volume settings (media, music, ringer, system, voice)

Location: longitude/latitude, accuracy, speed, altitude

Place: nearby place name (populated), most relevant city

Weather: wind direction, speed, clouds, temperature, dew point, humidity, air pressure

Ambient: light, proximity, temperature, pressure, noise, digital environment (WiFi and BT network information)

Activity: acceleration, user and device orientation, UI mode (undocked, car, desk), screen on/off, running apps

Player: artist, album, track name, track id, track length, genre, playback position, playlist name, playlist type, player state (repeat, shuffle mode), audio output (headset plugged)
Currently in Evaluation…

Which granularity/abstraction level to choose for representation/learning?

*user context data:*
  e.g., position as GPS coordinates or nearest agglomeration
  → data transformation/aggregation

*musical metadata:*
  artist/performer, album, track, genre, mood, other tags?
Conclusions

Music Information Retrieval is a broad and diverse field

Various approaches to extract information directly from the audio signal

Various sources and approaches to extract contextual data and similarity information from the Web

Multi-modal retrieval promising and allows for exciting applications

Next big challenges:
- modeling user context
- personalization
- situation-based retrieval
Thank you!

Latest version of the slides will be available at