JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

TNF
Technisch-Naturwissenschaftliche
Fakultät

Text-Based Description of Music for Indexing, Retrieval, and Browsing

DISSERTATION

zur Erlangung des akademischen Grades

Doktor

im Doktoratsstudium der

Technischen Wissenschaften

Eingereicht von:
Dipl.-Ing. Peter Knees

Angefertigt am:
Institut für Computational Perception

Beurteilung:
Univ.Prof. Dipl.-Ing. Dr. Gerhard Widmer (Betreuung)
Ao.Univ.Prof. Dipl.-Ing. Dr. Andreas Rauber

Linz, November 2010

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Kurzfassung

Ziel der vorliegenden Dissertation ist die Entwicklung automatischer Methoden zur Extraktion von Deskriptoren aus dem Web, die mit Musikstücken assoziiert werden können. Die so gewonnenen Musikdeskriptoren erlauben die Indizierung umfassender Musiksammlungen mithilfe vielfältiger Bezeichnungen und ermöglichen es, Musikstücke auffindbar zu machen und Sammlungen zu explorieren. Die vorgestellten Techniken bedienen sich gängiger Web-Suchmaschinen um Texte zu finden, die in Beziehung zu den Stücken stehen. Aus diesen Texten werden Deskriptoren gewonnen, die zum Einsatz kommen können

- zur Beschriftung, um die Orientierung innerhalb von Musikinterfaces zu vereinfachen (speziell in einem ebenfalls vorgestellten dreidimensionalen Musikinterface),
- als Indizierungsschlagworte, die in Folge als Features in Retrieval-Systemen für Musik dienen, die Abfragen bestehend aus beliebigem, beschreibendem Text verarbeiten können, oder
- als Features in adaptiven Retrieval-Systemen, die versuchen, zielgerichtete Vorschläge basierend auf dem Suchverhalten des Benutzers zu machen.

Im Rahmen dieser Dissertation werden verschiedene Strategien zur Extraktion von Deskriptoren, sowie zur Indizierung und zum Retrieval von Musikstücken erarbeitet und evaluiert. Weiters wird das Potenzial Web-basierter Retrieval-Ansätze, die um signalbasierte Ähnlichkeitsinformation erweitert werden, sowie das Potenzial Audioähnlichkeitsbasierter Suchansätze, die mit Web-Daten erweitert werden, untersucht und anhand von Prototypanwendungen demonstriert.

Abstract

The aim of this PhD thesis is to develop automatic methods that extract textual descriptions from the Web that can be associated with music pieces. Deriving descriptors for music permits to index large repositories with a diverse set of labels and allows for retrieving pieces and browsing collections. The techniques presented make use of common Web search engines to find related text content on the Web. From this content, descriptors are extracted that may serve as

- labels that facilitate orientation within browsing interfaces to music collections, especially in a three-dimensional browsing interface presented,
- indexing terms, used as features in music retrieval systems that can be queried using descriptive free-form text as input, and
- features in adaptive retrieval systems that aim at providing more user-targeted recommendations based on the user's searching behaviour for exploration of music collections.

In the context of this thesis, different extraction, indexing, and retrieval strategies are elaborated and evaluated. Furthermore, the potential of complementing Web-based retrieval with acoustic similarity extracted from the audio signal, as well as complementing audio-similarity-based browsing approaches with Web-based descriptors is investigated and demonstrated in prototype applications.

Acknowledgments

First and foremost thanks for everything are due to my family. Without them, this thesis would not be.

At least equally responsible for making this thesis possible is my supervisor Gerhard Widmer. I would like to thank him for giving me the opportunity to join his research group in Linz, for giving me space to work on this and other interesting topics, and for supporting me where ever possible and whenever necessary. Furthermore, I want to thank him for enabling me to visit many interesting conferences in amazing places and to meet the most extraordinary people. Also, I want to thank my second reader, Andreas Rauber, for his inspirational ideas and for providing support over the years.

I would like to thank all my co-authors and colleagues from the Department of Computational Perception and the OFAI in Vienna I got to work and be around with in the last years, namely Elias Pampalk, Tim Pohle, Markus Schedl, Dominik Schnitzer, Klaus Seyerlehner, Andreas Arzt, Arthur Flexer, Sebastian Flossmann, Harald Frostel, Martin Gasser, Werner Goebl, Maarten Grachten, Bernhard Niedermayer, Josef Scharinger, Claudia Kindermann, and, most recently, Reinhard Sonnleitner and Sebastian Böck. I also want to acknowledge all the students contributing to the projects I was involved in.

There are numerous people I have met during conferences that I had insightful discussions with and that have influenced my research. I can not and do not want to list them all here, but I would particularly like to mention Òscar Celma, Gijs Geleijnse, and Noam Koenigstein for the good cooperation and for promoting the field of Web-MIR. For all the others, if you think we had a good time, please feel acknowledged, as I had a good time too. Special credits for additional motivation to finish this thesis — unless not mentioned before — are due to Simon Dixon and KatieAnna Wolf.

This research was supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF) under project numbers L112-N04 ("Operational Models of Music Similarity for Music Information Retrieval") and L511-N15 ("Music Retrieval Beyond Simple Audio Similarity").

Contents

List of Tables

List of Figures

List of Abbreviations

CD	Compact Disc
CF	Collaborative Filtering
DCT	Discrete Cosine Transform
DF	Document Frequency
FFT	Fast Fourier Transformation
FP	Fluctuation Pattern
GMM	Gaussian Mixture Model
HITS	Hypertext Induced Topic Search
HTML	Hypertext Markup Language
IDF	Inverse Document Frequency
IE	Information Extraction
IR	Information Retrieval
ISMIR	International Society for Music Information Retrieval (since 2009)
LSA	Latent Semantic Analysis
MDM	Music Description Map
MDS	Multidimensional Scaling
MFCC	Mel Frequency Cepstral Coefficient
MIDI	Musical Instrument Digital Interface
MIR	Music Information Retrieval
MIREX	Music Information Retrieval Evaluation eXchange
MSA	Multiple Sequence Alignment
NLP	Natural Language Processing
NMF	Non-negative Matrix Factorization

P2P	Peer-to-Peer
PCA	Principle Components Analysis
PCM	Pulse Code Modulation
PoS	Part-of-Speech
RMS	Root Mean Square
RRS	Rank-based Relevance Scoring
RSS	Really Simple Syndication
SDH	Smoothed Data Histogram
SOM	Self-Organizing Map
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TF	Term Frequency
TSP	Travelling Salesman Problem
UI	User Interface
URL	Uniform Resource Locator

Chapter 1

Introduction

Music is everywhere. Music is for everyone. Music is more than just the pure acoustic perception, music is a pop cultural phenomenon – maybe even the most traditional and most persistent in human history. It takes a central role in most people's lives, whether they act as producers or consumers, and has the power to amplify or change its listener's emotional state. Even more, for many people, their musical preferences serve as a display of their personality. In short, if we deal with music, we must be aware that many factors have to be considered, more or less all of them far beyond the technical definition of sound as sensation of the ear stimulated by an oscillation of pressure (cf. [Wikipedia, 2010f]).

Given its cultural importance, it seems no wonder music was the first type of media that underwent the so-called digital revolution. Based on the technological advancements in encoding and compression of audio signals (most notably the invention of the mp3 standard) together with the establishment of the Internet as mainstream communication medium and distribution channel, and, in rapid succession, the development of high capacity portable music players, in the late 1990s, digital music has not only stirred up the IT industry, but also initiated a profound change in the way people "use" music. Today, a lot more people are listening to a lot more music in a lot more situations than ever before. Music has become a commodity that is naturally being traded electronically, exchanged, shared (legally or not), and even used as a means for social communication. Despite all these changes in the way music is *used*, the way music collections are *organised* on computers and music players and the way people *search* for music within these structures have basically remained the same.

Currently, the majority of systems for accessing music collections – irrespective of whether they comprise thousands (private collections) or millions of tracks (digital music resellers) – makes use of arbitrarily assigned and subjective meta-information like *genre* or *style* in combination with (nearly) objective meta-data like *artist name*, *album name*, *track name*, *record label*, or *year of release* to index the underlying music collection. Often, the hierarchical scheme *genre – artist – album – track* is then used to allow for browsing within the collection. While this may be sufficient for small private collections, in cases where most contained pieces are not known a-priori, the unmanageable amount of pieces may easily overstrain the user and impede the discovery of desired music. Thus, a person searching for music, e.g., a potential customer, must already have a very precise conception of the expected result. Obviously, the intrinsic problem of these indexing approaches is the limitation to a rather small set of meta-data, whereas the musical, or more general, the cultural context of music pieces is not captured. This results in inadequate representations and makes retrieval of desired pieces impractical and unintuitive.

As a response to these shortcomings of interfaces to music collections, the still growing but already well-established research field known as "Music Information Retrieval" (MIR) is — among others — developing methods that aim at extracting musical descriptors directly from the audio signal. Representations built upon these descriptors allow, for instance, for applications that autonomously analyse and structure music collections according to some of their *acoustic properties*, or systems that recommend *similar sounding* music to listeners based on the music they already own. While these signal-based approaches open up many opportunities for alternative music interfaces based directly on the audio content, they are unable to capture the non-acoustic, i.e., the contextual, aspects of music. Furthermore, as the descriptors derived from the audio signal usually consist of low-level features of the signal (as opposed to common high-level concepts, such as melody or rhythm) that are optimised to perform well in their application area, the obtained representations used to index the music collection have often no significance for humans.

1.1 Contributions

The objective of this thesis is to develop methods that allow for overcoming the limitations imposed by current music indexing strategies, i.e., those strategies that are based solely on a few categories of meta-data, as well as those based on audio content analysis alone. The fundamental idea behind all presented methods is to automatically exploit music-relevant information present on the Internet, which can be considered the central source of today's common knowledge, to derive *characteristic descriptions* for music pieces and representations of their *cultural context*. More precisely, for a given collection of music pieces, Web pages about these pieces are retrieved via a common Web search engine. In general, it is assumed that these pages contain relevant information about the respective music pieces or the work of the corresponding artists. Thus, relations between textual descriptions and music pieces are established. Furthermore, by extracting information from the texts, human understandable representations of music that make use of a rich vocabulary are obtained. These representations can facilitate access and improve interaction when used for labelling, indexing, and retrieval. In combination with audio-based similarity approaches, Web-based characterisations can even yield additional benefits. This is shown in this thesis by elaborating on three concepts that rely on text-based descriptions to access music collections. For demonstration purposes, all three concepts are also realised in prototype interfaces. A schematic overview of the contributions of this thesis can be seen in Figure 1.1.

In the first concept (the blue bar on the left in Figure 1.1), text descriptors from the Web are incorporated for augmentation of so-called Music Maps, i.e., two-dimensional graphical arrangements of music collections, such that similar sounding music pieces are placed closely together, whereas dissimilar pieces are located far apart. While an arrangement in this fashion is usually very intuitive, it is difficult to

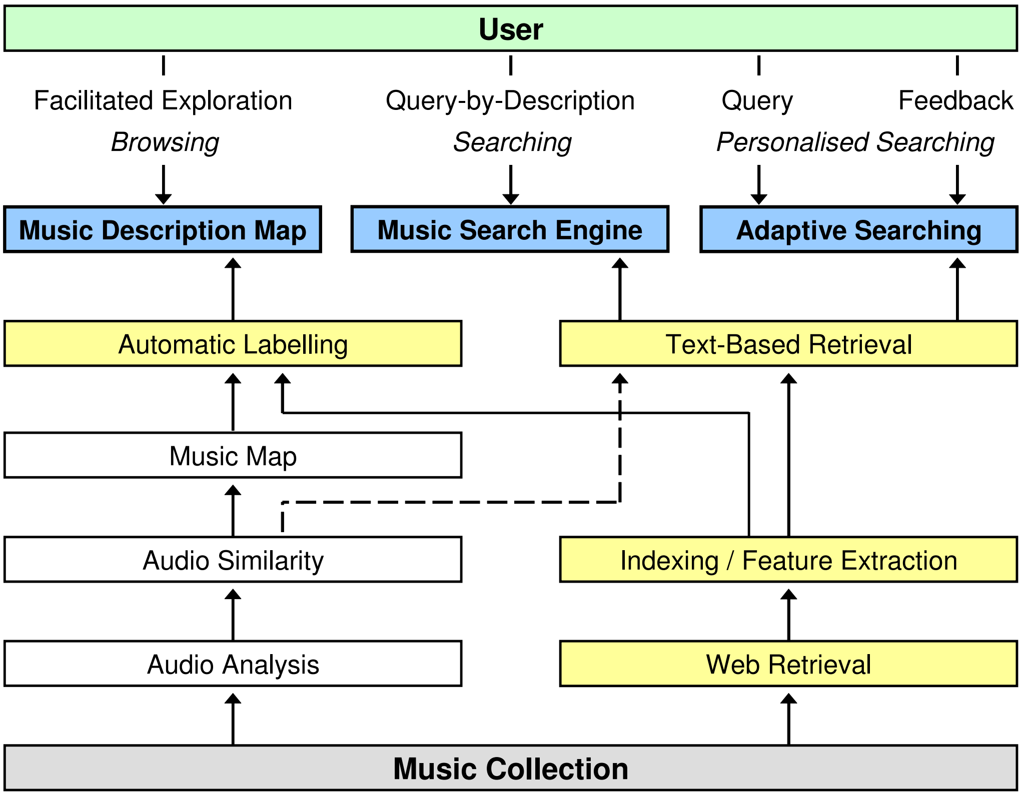


Figure 1.1: Schematic overview of the methods presented in this thesis. Bars highlighted in yellow represent the main methods elaborated on in this thesis. The three blue bars represent the central contributions that make use of the underlying information: the Music Description Map for facilitated browsing, the Music Search Engine for retrieving music from collections via descriptive queries, and the Adaptive Searching approach that exploits user feedback to allow for targeted searching and browsing.

assess which type of music is to be found in which regions without having to listen to them in advance. Additionally, large Music Maps tend to be cluttered with the labels of the contained tracks. Here, descriptions automatically derived from Web-data can be utilised for labelling coherent regions on the maps. In the technique called *Music Description Map (MDM)*, musically-relevant terms serve as landmarks and allow for better orientation. For instance, instead of having many overlapping labels of tracks from *Miles Davis* and other similar sounding pieces, the MDM displays musical descriptions of this type of music such as “Trumpet” and “Jazz”. The idea of creating characteristic landmarks for augmented navigation is further applied in the *nepTune* interface, where Music Maps are raised to the third dimension to allow interactive exploration of a virtual reality landscape created from a music collection.

The principle of matching music pieces with their cultural context is further developed and deepened in the second concept (the blue bar in the centre in Figure 1.1). Instead of presenting the indexing keywords to the user to facilitate browsing, representations can be compared to natural language queries. This opens up the possibility to build a *Music Search Engine* that can be used like every Web search engine, i.e., in today's most common and most accepted manner for searching: by typing arbitrary keywords. For example, instead of just finding tracks that are labelled as "Rock", a query like "Rock with Great Riffs" can be formulated to emphasise the importance of energetic guitar phrases. Another query could be "Chicago 1920" to express the intention of finding Jazz pieces originating from this particular area and time. With the integration of acoustic similarity, also tracks without contextual information present on the Web can be included into the retrieval process.

The third concept (the blue bar on the right in Figure 1.1) aims at combining the derived descriptors with usage feedback to develop more personalised and user-oriented music services. As with the music search engine, the process of *Adaptive Searching* starts with submitting a descriptive query to retrieve an initial set of results. By browsing through that set and selecting pieces of interest, other pieces that could be after the user's fancy are proposed. Using this iterative process, targeted searching and assisted exploration of the collection is facilitated. For all three of the presented concepts, the goal is to demonstrate the potential of Web-based music descriptors for developing methods that support those users who are actively searching for music they might like, in contrast to passive consumers who expect automatically delivered music recommendations based on what they already listen to.

1.2 Why Automatic Extraction from the Web?

One of the central tasks of this thesis is the labelling (or indexing) of large music collections with "semantic" descriptors. In this context, the somewhat misleading notion of "semantic" refers to "words that make sense to humans when describing music" and that are therefore of help to other humans that search or browse for music. Furthermore, having a set of words for indexing, computers can be used to assist in these tasks.

However — ignoring the fact that such labels are always subjective — labelling of very large music collections (on the order of millions of tracks) is a very labour-intense task. On the other hand, well-labelled music repositories are valuable, not least from an economical point of view. Not surprisingly, it is of interest to efficiently or even automatically accomplish such tasks, for instance by developing methods that permit a computer to learn how to find or derive such descriptions for music (e.g., as proposed in the present thesis). This approach can be classified as belonging to the broad field of *artificial intelligence*. Public interest in artificial intelligence has varied considerably in the last decades, with the last buzz probably due to "intelligent algorithms" that bring order to the Web, i.e., Web search engines, especially *Google*[1]. Following the evolution of the Web into a social medium, it seems recently a trend towards collaborative approaches has established instead. Since automatic, "intelligent" methods are limited and may not satisfy the user fully, tasks such as organising (and labelling) the data on the Web or recommending content are increasingly done by (trusted) humans to provide results closer to "what humans want". The general process of exploiting the capacities of communities and solving large scale problems in a distributed manner, usually organised over the

[1] `http://www.google.com`

Web, is frequently also referred to as *crowdsourcing*. Examples of tasks that can be performed efficiently by crowdsourcing are labelling of images, e.g., by playing a game (and contributing unconsciously, cf. the *ESP game* by [von Ahn and Dabbish, 2004]), tagging of music pieces and artists (cf. *Last.fm*[2]), or the development of open source operating systems.

This thesis is dedicated to contributing to the field of artificial intelligence rather than promoting methods for distributed collaboration for the following reasons. While crowdsourcing approaches have the potential to be applied to a wide area of problems and are useful for accomplishing otherwise expensive tasks quickly and efficiently, they are limited in that they still require human manpower. Usually this also entails the necessity of addressing and finding people willing to participate — unless they are unaware of participation as when playing games (see above). From an ethical point of view, this kind of exploitation of participants might be problematic. Furthermore, open crowdsourcing projects, such as tagging of music as implemented by Last.fm, are prone to suffer from effects such as a "community bias" or a "popularity bias" because most people help contributing tags to the same few popular tracks or artists, whereas lesser known artists (artists from the so-called "long-tail", cf. [Anderson, 2006]) are usually neglected by the community. Last but not least, in the opinion of the author, research on "intelligent algorithms" is much more interesting and challenging than solving problems by crowdsourcing. Therefore, the present thesis deals with the development of methods to automatically extract descriptions for music from the Web.

1.3 Organisation of this Thesis

The remainder of this work is organised as follows. Chapter 2 gives an overview of related work from the fields of Music Information Retrieval, Web Information Retrieval, Multimedia, and User Interfaces. In Chapter 3, the methodological foundations of this work are explained. Chapter 4 then elaborates the concept of augmenting interfaces to music collections by labelling with automatically extracted descriptions. Furthermore, the interface prototype nepTune is presented. Chapter 5 introduces different approaches for constructing a music retrieval system that captures cultural information and is capable of dealing with diversified text queries. Chapter 6 presents steps towards personalisation and user-targeted music retrieval systems that make use of explicit feedback. The thesis concludes with a critical discussion and an outlook to future trends and developments (Chapter 7).

[2] `http://www.last.fm`

Chapter 2

Related Work

The aim of this chapter is to review related work, i.e., methods and applications from the fields of Music Information Retrieval and Information Retrieval (IR). The first section (2.1) puts an emphasis on work dealing with music-related feature extraction from both content and contextual data. Feature extraction, i.e., calculation of characteristic descriptors for entities, is a crucial task as it enables indexing of individual entities as well as calculation of similarity between entities. The concept of similarity generally plays a central role throughout this chapter. For example, in Section 2.1.2 – which deals with text-based indexing and retrieval and their most prominent applications as (Web) search engines – relevance of an indexed document to a query can be obtained by calculating the similarity of their respective representations. After presentation of related methods, the focus is shifted to related applications. Section 2.3 reviews existing approaches to music search engines. Finally, Section 2.4 deals with user interfaces to music collections that facilitate browsing and active search for new and interesting music.

2.1 Music Similarity and Indexing

Since this thesis is situated within an MIR-context, naturally, most related work originates from this field. However, it should be stated that from a strictly chronological point of view, the more generic field of IR (cf. Section 2.2) would have to be introduced first. For now – and to clarify the relation of the two fields – the definition of IR as "science of searching for documents, for information within documents, and for metadata about documents" [Wikipedia, 2010b] is sufficient. In practice, in IR these documents consist of texts. Correspondingly, MIR is the multidisciplinary science of retrieving information from and about music.

MIR as a dedicated research field has evolved in the 1990's and been fully established with the organisation of the ISMIR conference series [Byrd and Fingerhut, 2002]. The field comprises of a variety of topics and is hence probably most comprehensively defined extensionally, for example through the list of topics to be found in the Call-for-Papers of ISMIR 2010[1]. The following excerpt of this list comprises the ten topics most related to this thesis:

- content-based querying and retrieval

[1] `http://ismir2010.ismir.net/information-for-authors/call-for-papers/`

- music recommendation and playlist generation
- music signal processing
- database systems, indexing and query languages
- text and web mining
- evaluation of MIR systems
- knowledge representation, social tags, and metadata
- genre, style and mood
- similarity metrics
- user interfaces and user models

Besides this, also topics such as music perception, optical music recognition, extraction and modelling of higher musical concepts such as melody or rhythm, or automatic real-time score following, to name but a few, are central research questions. In the following, work that deals with music representations suitable for indexing, retrieval, and organisation of collections is addressed. Furthermore, as can be seen from the titles in this section, only information obtained directly from an audio signal or from sources that resemble aspects of the cultural context of music are taken into consideration. Sheet music as well as symbolic notation languages, e.g., MIDI scores, are excluded. However, in Section 2.3, also music search engines that are based on symbolic music representations are discussed.

2.1.1 Content-Based Similarity

The aim of content-based approaches is to extract information directly from the audio signal, more precisely from a digital representation of a recording of the acoustic wave, for example encoded as PCM. The basic approach to feature extraction is as follows (cf. Figure 2.1): The audio signal is typically chunked into a series of short segments called frames. Optionally, each frame can be transformed from the time-domain representation to a frequency-domain representation using an FFT. Thereafter, feature extraction is performed on each frame. From the derived features, a model that summarises the extracted frame-level features can be used as representation of the track. These models can then, for instance, be utilised for calculating pairwise similarities of audio tracks. This information is essential when retrieving similar music pieces (e.g., in automatic recommendation or playlist generation, i.e., *query-by-example* scenarios), for automatic genre classification (e.g., [Tzanetakis and Cook, 2002, Aucouturier and Pachet, 2003]), or to automatically structure and organise music collections according to acoustic closeness.

For the feature extraction step, signal properties of interest range from *low-level features*, i.e., features that describe or abstract aspects directly from the signal or its frequency-domain representation, to *high-level features*, i.e., features that describe musically "meaningful" concepts (cf. [Casey et al., 2008]). Low-level features commonly found in the literature are, for instance, Zero Crossing Rate, Spectral Centroid, Spectral Flux, RMS, Pitch-Class Profiles (frequently also referred to as

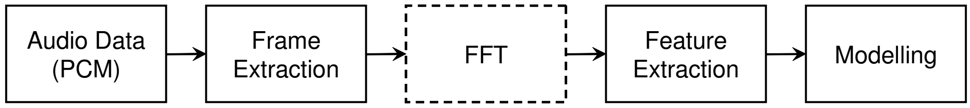


Figure 2.1: Generic overview of the audio feature extraction process.

Chroma Vectors), Fluctuation Patterns, and MFCCs. A comprehensive comparison and evaluation of these and other measures can be found in [Pohle, 2005]. High-level features comprise musical properties such as rhythm, melody, pitch, harmony, structure, or lyrics and are difficult to extract. In between, often also a category of *mid-level features* is defined, comprising features that are built from low-level features by incorporating musical knowledge with the aim of capturing properties related to high-level features (e.g., [Bello and Pickens, 2005, Marolt, 2006]). Boundaries between these categories are not always clear, hence it is sometimes debatable for features to which level of abstraction they belong to.

The content-based similarity measure utilised in this work is built upon algorithms that capture spectral properties by exploiting MFCCs, as well as on algorithms aiming at extracting rhythmic patterns, or better, patterns of periodicity, i.e., the so-called Fluctuation Patterns. A technical description of both approaches can be found in Section 3.2. For a comprehensive overview of other signal-based features and their specific applications, the reader is referred to [Casey et al., 2008].

2.1.2 Context-Based Indexing and Similarity

As already mentioned in the introduction of this work, when trying to model music similarity algorithmically, the *cultural context* should not be neglected as it contains important aspects not directly included in the audio signal. On one hand, this is motivated by the fact that there is no a-priori valid definition of what makes two musical entities similar. (Is it the melody, the instrumentation, the tempo, or the fact that two artists share certain political views?) On the other hand, modelling of contextual factors gains more and more importance as similarity measures relying solely on content analysis have reached a "glass-ceiling" in terms of classification accuracy [Aucouturier and Pachet, 2004].

In this section, work that deals with *music-related* data and extraction of interesting musical aspects not found in the signal is reviewed. Although the presented methods and their intended applications vary considerably, they all have in common that they make use of some form of contextual data – often also referred to as "cultural features", "community meta-data", or "context-based features". Irrespective of its name, such kind of data originates from external sources (in the following sections from sources that can be accessed primarily through Web technology) and covers some of the cultural facets that influence the human perception of music.

Incorporating context-based information permits, for example, automatic tagging of artists or music pieces (e.g., [Eck et al., 2007, Sordo et al., 2007]) — also referred to as "semantic indexing" (e.g., [Whitman, 2005, Turnbull et al., 2007a]), automatic biography generation (e.g., [Alani et al., 2003]), enriching music players with meta-information (e.g., [Schedl et al., 2006b]), to enhance user interfaces to mu-

	Content-Based	**Context-Based**
Prerequisites	music file	meta-data
Popularity bias	no	yes
Features	objective	subjective
	direct	noisy channel
	low-level	high-level

Table 2.1: A comparison of content- and context-based feature properties.

sic collections (cf. Section 2.4), or simply to estimate the similarity of two musical entities. As with content-based similarity measures, application fields for context-based similarity measures are manifold. For example, they can be used for automatic music recommendation (cf. [Celma and Lamere, 2007]), automatic playlist generation (e.g., [Aucouturier and Pachet, 2002c, Pohle et al., 2007a]), to unveil artist relationships (e.g., [Cano and Koppenberger, 2004]), or to build music search engines (cf. Section 2.3). A comprehensive overview of context-based approaches for similarity assessment can be found in [Schedl and Knees, 2009]. Table 2.1 aims at giving a brief comparison of content- and context-based feature properties.

Before examining approaches from the literature in detail, general implications of incorporating context-based similarity measures are discussed (cf. [Turnbull et al., 2008a]). Usually, and in contrast to content-based features, to obtain context-based features it is not necessary to have access to the actual music file. By having a list of artists, applications like, for instance, music information systems can be built without any acoustic representation of the music under consideration as in [Schedl, 2008]. On the other hand, without meta information like *artist* or *title*, context-based approaches are inapplicable. Improperly labelled pieces and ambiguous identifiers also pose a problem (cf. [Geleijnse and Korst, 2007]). Furthermore, all cultural methods depend on the existence of available meta-data, i.e., music not present within the respective sources is virtually inexistent. This may be the case for music from the so-called "long-tail" (cf. [Anderson, 2006]), i.e., lesser known or special interest music ("popularity bias"), as well as for up-and-coming music and sparsely populated (collaborative) data sources ("cold start problem"), cf. [Celma, 2008]. To sum up, the crucial point is that in order to derive contextual features, one must have access to a large amount of user generated data. Assuming this condition can be met, community data provides a rich source of information on the social context and reflects the "collective wisdom of the crowd" without any explicit or direct human involvement necessary. Furthermore, using contextual features, it is possible to model and monitor temporal changes. By relying on constantly updated sources, cultural features are capable of capturing recent developments and emerging trends. This is especially important in the music domain, where the public perception of artistic work can vary significantly over time. Table 2.2 gives an overview of some aspects of the different context-based approaches presented in this section.

	Manual	**Tags**	**Web-Terms**	**Lyrics**
Source	experts	Web service	Web pages	lyrics portals
Community Req.	no	yes	depends	no
Level	all	artist, track	artist	track (artist)
Feature Dim.	depends	moderate	very high	possibly high
Specific Bias	expert	community	low	none
Potential Noise	none	moderate	high	low

	Co-Occ.	**Playlists**	**P2P**	**CF**
Source	Web (search)	radio, CDs, Web	shared folders	users
Community Req.	no	depends	yes	yes
Level	artist	artist, track	artist, track	all
Feature Dim.	sim. matrix	sim. matrix	sim. matrix	user-item-mat.
Specific Bias	low	low	community	community
Potential Noise	high	low	high	yes

Table 2.2: Overview of different context-based approaches.

2.1.2.1 Manual Annotations

The most intuitive approach to obtain meta-data for music is simply by asking humans. Since most high-level categories can not be sufficiently modelled algorithmically, it seems straightforward to rely on experts judgements for semantic labelling. Two examples of such efforts are *allmusic*[2] and the *Music Genome Project*[3]. While allmusic is more focused on editorial meta-data, the Music Genome Project aims at representing songs in a feature space consisting of about 400 musically relevant dimensions, i.e., semantic descriptors called "genes" [Wikipedia, 2010d]. According to reports, these descriptors comprise very detailed and specific attributes such as "Emphasis on Varied Instrumentation", "Easy Listening Qualities", "Hardcore Rap Influence", "Prominent Saxophone Part", "Sexist Lyrics", or "Breathy Vocals" [Wikipedia, 2010c]. The song representations are used within the Web-streaming service *Pandora*[4] to provide listeners with high quality recommendations. However, manual annotation is a very labour-intensive task and evaluation of a single song by a musician takes approximately 20-30 minutes. Hence, it is obvious that the vast number of (commercial) music pieces available is highly unlikely ever to be fully annotated by musically trained experts. Another aspect concerns the objectivity of expert judgements. Although description categories aim at capturing properties that are objectively either present or not, attributes like "Interesting Song Structure" or "Great Piano Solo" inevitably are inherently subjective. To compensate for expert's biases, multiple opinions may be included — which further increases annotation costs. Finally, subjectivity is even more pronounced for expert-generated content such as reviews and comments.

[2] `http://www.allmusic.com/`, formerly *All Music Guide (AMG)*.
[3] `http://www.pandora.com/mgp.shtml`
[4] `http://www.pandora.com`

Figure 2.2: Last.fm tag cloud for the band *Daft Punk*.

2.1.2.2 Collaborative Tags

Another approach to collect semantic annotations is to exploit knowledge and manpower of Web communities. As one of the characteristics of the so-called "Web 2.0" — where Web sites encourage (even rely on) their users to participating in the generation of content — available items such as photos, films, or music can be labelled with tags by the user community. A tag can be virtually anything, but it usually consists of a short description of one aspect typical to the item (for music, for example, genre or style, instrumentation, mood, or performer). The more that people label an item with a tag, the more that tag is assumed to be relevant to the item. For music, the most prominent platform that makes use of this approach is Last.fm. Figure 2.2 shows an example of user-generated tags on Last.fm, i.e., the most important tags assigned to the French electronic band *Daft Punk*. Since Last.fm offers access to the collected tags in a standardised manner, it is a very valuable source of context-related information.

Using tag profiles of artists or tracks, an estimation of similarity can be obtained by calculating a distance or a tag overlap. Furthermore, several approaches make use of tags to collect music-related terms to build a feature vocabulary for related tasks, e.g., [Pohle et al., 2007b] and [Hu et al., 2009]. [Levy and Sandler, 2008] retrieve tags from Last.fm and *MusicStrands*[5] (a conceptually similar, but in the meantime discontinued Web service), to construct a semantic space for music pieces. To this end, all tags found for a specific track are processed like text documents and a standard TF·IDF-based document-term matrix is created, i.e., each track is represented by a term vector (see Section 3.1.3). Different calculation methods are explored, including dimensionality reduction by applying Latent Semantic Analysis (LSA) [Deerwester et al., 1990]. Following a similar approach, [Laurier et al., 2009] explicitly address the usage of tags for music mood representation.

In a broader context, collaborative tagging can be seen as a (more democratic) extension to the rather elitist approach of expert annotations discussed in Section 2.1.2.1. That is, instead of relying on individual opinions, an "average" opinion is admitted. Additionally, an enthusiastic community is usually faster in annotating large corpora. Further advantageous results of tag-based approaches are a music-targeted and small, yet unrestricted, vocabulary with a tolerable amount of unrelated terms ("noise") and availability of descriptors for individual tracks (in contrast to, for example, most Web-term-based approaches, cf. Section 2.1.2.3). Conversely, tag-based approaches also suffer from limitations. For example, without a large and active user community, sufficient tagging of comprehensive collections is infea-

[5] was: `http://www.musicstrands.com`, now: `http://www.strands.com`

sible. Furthermore, tracks from the so-called “long-tail”, i.e., less popular tracks, are usually only very sparsely tagged and — if the community is too homogeneous — effects such as a “community bias” may be observed. Another serious issue is the vulnerability to erroneous information, be it introduced by mistake or purpose (hacking). An attempt to filter noisy and redundant tags is presented by [Geleijnse et al., 2007]. By comparing tags of an artist with the set of tags associated with tracks by that artist, a more reliable and consistent annotation scheme is generated.

To improve and speed up the annotation process, gathering tags via games has become very popular, e.g., [Law et al., 2007], [Mandel and Ellis, 2007], and [Turnbull et al., 2007b]. Initially developed to improve tagging in the image domain [von Ahn and Dabbish, 2004], such games provide some form of incentive — be it just the pure joy of gaming — and motivate the player to engage in tasks such as finding precise descriptors for music pieces. By encouraging users to play such games, a large number of songs can be efficiently annotated with semantic descriptors. Another recent trend to alleviate the data sparsity problem is automatic tagging/propagation of tags based on alternative (usually content-based) data sources, e.g., [Sordo et al., 2007, Eck et al., 2008, Kim et al., 2009].

2.1.2.3 Web-Text Term Profiles

Possibly the most extensive source of cultural data are the zillions of Web pages available on the Internet. To make the valuable music-relevant information embedded within this huge pool accessible, the majority of the presented approaches uses a Web search engine to retrieve related documents. In order to restrict the search to Web pages relevant to music, different query schemes are proposed. For instance, such schemes may comprise the artist’s name augmented by the keyword sequence *music review* [Whitman and Lawrence, 2002, Baumann and Hummel, 2003] or *music genre style* [Knees et al., 2004]. Additional keywords are particularly important for artists whose names have another meaning outside the music context, such as *50 Cent*, *Hole*, and *Air*. A comparison of different query schemes can be found in [Knees et al., 2008b]. Using the (unstructured) Web texts, tasks such as music/artist similarity estimation and genre (or more general, label) prediction can be modelled as traditional IR problems. Thus, a variety of established and well-explored techniques can be applied, e.g., a *bag-of-words* model in conjunction with a TF·IDF-weighting to create artist-specific term profiles (cf. Section 3.1.3).

In seminal work, [Whitman and Lawrence, 2002] extract different term sets (unigrams, bigrams, noun phrases, artist names, and adjectives) from up to 50 artist-related pages obtained via the Web search engine Google. After downloading the Web pages, the authors apply parsers and a Part-of-Speech (PoS) tagger [Brill, 1992] to determine each word’s part of speech and the appropriate term set. Based on term occurrences, individual term profiles are created for each artist by employing a TF·IDF-weighting variant, which assigns a weight to each term in the context of each artist. The general idea of TF·IDF is to consider terms that occur often within the document (here, the Web pages of an artist), but rarely in other documents (other artists’ Web pages). Technically speaking, terms that have a high *term frequency* (TF) and a low *document frequency* (DF) or, correspondingly, a high *inverse document frequency* (IDF) are assigned higher weights (cf. Section 3.1.3). Alternatively, the authors propose another variant of weighting in which rarely occurring terms, i.e., terms with a low DF, should also be weighted down to emphasise terms in the middle IDF range. This scheme is applied to all term sets except for adjectives. Calculating the TF·IDF weights for all terms in each term set yields individual feature vectors or term profiles for each artist. The *overlap* between the term profiles of two artists, i.e., the sum of weights of all terms that occur in both artists' sets, is then used as an estimate of their similarity. For evaluation, the authors compare these similarities to two other sources of artist similarity information, which serve as ground truth (similar-artist-relations from the All Music Guide - now allmusic - and user collections extracted from the Peer-to-Peer Network *OpenNap*, cf. Section 2.1.2.8).

Extending the work presented in [Whitman and Lawrence, 2002], [Baumann and Hummel, 2003] introduce filters to prune the set of retrieved Web pages. They discard all Web pages with a size of more than 40kB after parsing and ignore text in table cells if it does not comprise at least one sentence and more than 60 characters to exclude advertisements. Finally, they perform keyword spotting in the URL, the title, and the first text part of each page. Each occurrence of the initial query constraints (i.e., the artist's name, as well as the terms *music* and *review*) contributes to a page score. Pages with a low score are filtered out. In contrast to Whitman and Lawrence, Baumann and Hummel use a logarithmic IDF weighting in their TF·IDF formulation. Using these modifications, the authors are able to outperform the approach presented by Whitman and Lawrence in terms of similarity prediction.

In [Knees et al., 2004], an approach is presented that applies similar Web mining techniques for the task of automatic artist-to-genre classification. For Web data retrieval, Google and *Yahoo!*[6] are compared. In contrast to [Whitman and Lawrence, 2002], only one unigram-based term list is constructed per artist. To this end, a TF·IDF variant is employed to create weighted term profiles (represented as vectors). Furthermore, the χ^2-test (Equation 6.2) is applied for term selection, i.e., to filter out terms that are less important to describe certain genres and to remove noisy dimensions in order to increase classification accuracy [Yang and Pedersen, 1997]. Note that for similarity computation, the χ^2-test can not be applied, as it relies on class assignments (in this case, genre information used for training) which are in general not available in this scenario. However, to calculate the similarity between the term profiles of two artists, the cosine similarity can be calculated on the unpruned term vectors. For classification, k-Nearest Neighbours (k-NN, e.g., [Cover and Hart, 1967]) and Support Vector Machines (e.g., [Vapnik, 1995]) are used. In [Knees et al., 2008b], the classification approach is re-examined. Not surprisingly, it is shown that genre classification mainly depends on the occurrence of proper names (typical artists, album titles, etc.) on Web pages. Table 2.3 shows the 100 highest-scoring terms from the genre "Jazz" according to the χ^2-test (performed on the *uspop2002* collection [Berenzweig et al., 2003]). Motivated by these insights, a simplified genre classification approach is proposed that does not require to download the top-ranked Web pages for analysis. Instead of extracting terms from Web pages relevant to a query like *"artist" music genre style*, Google is queried with a scheme like *"artist" "similar artists"* to enforce the occurrence of highly distinctive

[6] `http://www.yahoo.com`

duotones	teo	saxophonist	mobley	balakrishna
nefertiti	gil	mulligan	trumpeter	dameron
adderley	bartz	pangaea	blakey	flagelhorn
relaxin	saxophonists	concierto	cookin	ife
agharta	tingen	bess	alton	leu
songbird	modal	bitches	wynton	melonae
silhouette	lechafaud	breathless	porgy	mikkelborg
gorelick	macero	amandla	najee	palle
konitz	jarrett	frelimo	bop	saeta
tutu	cannonball	stitt	kilimanjaro	sitarist
eckstine	orea	harmon	improvisations	sivad
filles	grover	adderly	airto	shorter
nonet	kenny	airegin	sidemen	blackhawk
steamin	mabry	cleota	cosey	soprano
sketches	mtume	lascenseur	prestige	dewey
decoy	ascenseur	milesdavis	sanborn	miles
brew	boplicity	szwed	bebop	charlap
dejohnette	siesta	tadd	davis	sharrock
zawinul	aranjuez	yesternow	albright	birdland
lorber	freeloader	liebman	badal	ipanema

Table 2.3: Top 100 χ^2-ranked terms for the genre "Jazz" [Knees et al., 2008b].

proper names relevant to the genre. Since Google delivers not only links to relevant Web pages but also a short textual summary containing each result's most relevant section ("snippets"), for term extraction, only the Google result page is required.

Term profiles, as created in [Knees et al., 2004], are also used by [Pampalk et al., 2005] for hierarchical clustering of artists. Instead of constructing the feature space from all terms contained in the downloaded Web pages, a manually assembled vocabulary of about 1,400 terms related to music (e.g., genre and style names, instruments, moods, and countries) is used. For genre classification, the unpruned term set outperforms the vocabulary-based method.

Another Web-term-based approach is presented by [Pohle et al., 2007b]. Based on a data set of 1,979 artists gathered from allmusic, a vocabulary of about 3,000 tags is extracted from Last.fm and is used to create TF·IDF vectors from occurrences of these tags on artist-related Web pages. The authors then cluster the artists using Non-negative Matrix Factorization (NMF) [Lee and Seung, 1999] on the TF·IDF features. As NMF yields a weighted affinity of each artist to each cluster, Pohle et al. propose to use this approach for exploring the artist collection by controlling the weights of the resulting clusters (see also Section 2.4.3).

There further exist other approaches that derive term profiles from more specific Web resources. For example, [Celma et al., 2006] propose to crawl audio blogs via RSS feeds to calculate song-specific TF·IDF vectors, e.g., for usage in a music search engine (cf. Section 2.3). [Hu et al., 2005] extract TF-based features from music reviews gathered from *Epinions.com*[7].

2.1.2.4 Web-based Music Information Extraction

Instead of just providing features for music entities for similarity estimation or prediction of genre or other categories, Web pages can serve also as source of explicit meta-data and relation extraction. To this end, methods from Natural Language Processing (NLP), or more precisely, Information Extraction (IE), are applied, i.e.,

[7] `http://www.epinions.com`

methods that exploit the morphology of words and the syntax of human languages to (partially) access and model the contained information.

[Schedl and Widmer, 2007] aim at automatically identifying members of bands and their respective roles (played instruments) within a band. In this respect, they examine Web pages to detect patterns such as *M plays the I* or *M is the R*, where M represents a potential band member, I an instrument, and R a role. [Krenmair, 2010] investigates the usage of the GATE framework (see [Cunningham et al., 2002]) to automatically identify artist names and to extract band-membership relations and released albums/media for artists. [Schedl et al., 2010b] propose different approaches to determine the country of origin for a given artist. In one of these approaches, keyword spotting for terms such as "born" or "founded" is performed in the context of country names. [Geleijnse and Korst, 2006] use patterns like *G bands such as A*, *for example* A_1 *and* A_2, or *M mood by A* (where G represents a genre, A an artist name, and M a possible mood) to unveil genre-artist, artist-artist, and mood-artist relations, respectively.

2.1.2.5 Song Lyrics

The lyrics of a song represent an important aspect of the semantics of music since they usually reveal information about the "meaning" of a song, its composer, or the performer: e.g., cultural background (via different languages or use of slang words), political orientation, or style of music (use of a specific vocabulary in certain music styles). While automatic extraction of the lyrics directly from the audio signal is a very challenging and still unsolved task, lyrics for a good portion of available commercial songs can be found online. [Knees et al., 2005] and [Korst and Geleijnse, 2006] propose approaches to automatically retrieve lyrics that are as correct as possible (with respect to what is sung) from the Web by comparing and combining multiple sources.

[Logan et al., 2004] use song lyrics for tracks by 399 artists to analyse and compare the semantic content of the lyrics and to determine artist similarity. In a first step, Probabilistic Latent Semantic Analysis (PLSA) [Hofmann, 1999] is applied to a collection of over 40,000 song lyrics to extract N topics typical to lyrics. Second, all lyrics by an artist are processed using each of the extracted topic models to create N-dimensional vectors of which each dimension gives the likelihood of the artist's tracks belonging to the corresponding topic. Artist vectors are then compared by calculating the L_1 distance (also known as Manhattan distance). This similarity approach is evaluated against human similarity judgements, i.e., the "survey" data for the *uspop2002* set (see [Berenzweig et al., 2003]), and yields worse results than similarity data obtained via acoustic features (irrespective of the chosen N, the usage of stemming, or the filtering of lyrics-specific stopwords). However, as lyrics-based and audio-based approaches make different errors, a combination of both is suggested.

[Mahedero et al., 2005] demonstrate the usefulness of lyrics for four important tasks: language identification, structure extraction (i.e., recognition of intro, verse, chorus, bridge, outro, etc.), thematic categorisation, and similarity measurement. For similarity calculation, a standard TF·IDF measure with cosine distance is proposed as an initial step. Using this information, a song's representation is obtained by concatenating distances to all songs in the collection into a new vector. These representations are then compared using an unspecified algorithm. Exploratory experiments indicate potential for cover version identification and plagiarism detection.

Other approaches aim at revealing conceptual clusters (e.g., [Kleedorfer et al., 2008]) or at classifying songs into genres or mood categories. For instance, the objective of [Laurier et al., 2008] is classification of songs into four mood categories by means of lyrics and content analysis. For lyrics, the TF·IDF measure with cosine distance is incorporated. Optionally, LSA is applied to the TF·IDF vectors (achieving best results when projecting vectors down to 30 dimensions). Audio-based features perform better compared to lyrics-based features, however, a combination of both yields the best results. [Hu et al., 2009] experiment with TF·IDF, TF, and Boolean vectors and investigate the impact of stemming, PoS tagging, and function words for soft-categorisation into 18 mood clusters. Best results are achieved with TF·IDF weights on stemmed terms. An interesting result is that in this scenario, lyrics-based features alone can outperform audio-based features. Besides TF·IDF and PoS features, [Mayer et al., 2008] also propose the use of rhyme and statistical features to improve lyrics-based genre classification. To extract rhyme features, lyrics are transcribed to a phonetic representation and searched for different patterns of rhyming lines (e.g., AA, AABB, ABAB). Features consist of the number of occurrences of each pattern, the percentage of rhyming blocks, and the fraction of unique terms used to build the rhymes. Statistical features are constructed by counting various punctuation characters and digits, and calculating typical ratios like average words per line or average length of words. Classification experiments show that the proposed style features and a combination of style features and classical TF·IDF features outperform the TF·IDF-only-approach.

[Hirjee and Brown, 2009] analyse lyrics for rhyming style information to automatically identify (possibly imperfect) internal and line-ending rhymes. In continuation of this work (i.e., in [Hirjee and Brown, 2010]), the proposed high-level rhyme features can be used to identify the corresponding rap artist based on the stylistic information extracted from the lyrics.

In summary, recent literature demonstrates that many interesting musical aspects can be covered by exploiting lyrics information. However, since new and ground breaking applications for this kind of information have not been discovered yet, the potential of lyrics analysis is currently mainly seen as a complementary source to content-based features for genre or mood classification.

2.1.2.6 Web-based Co-Occurrences and Page Counts

Circumventing the necessity of creating feature profiles, the work reviewed in this and in the next two sections applies a more direct approach to estimate relations and/or similarity. In principle, the idea is that the occurrence of two music pieces or artists within the same context is considered to be an indication of some sort of similarity. As a source for this co-occurrence analysis, this section discusses Web pages and — as an abstraction — page counts returned by search engines.

Using music-related Web pages as data source for MIR tasks was potentially first performed by [Cohen and Fan, 2000]. Co-occurrences of artists are included in a Collaborative Filtering system which is used for artist recommendation (cf. Section 2.1.2.9). To determine pages relevant to the music domain, Cohen and Fan query the Web search engines *Altavista*[8] and *Northern Light*[9]. The resulting HTML pages are parsed and the plain text content is analysed for occurrences of entities, i.e., artists' names. Pages with multiple occurrences of artists are interpreted as "pseudo-users" within the collaborative system that "rate" all tracks by the contained artists positively. As one ground truth for evaluating their approach, Cohen and Fan implicitly propose another form of context-based similarity estimation, namely the exploitation of server logs (cf. Section 2.1.2.10).

A similar approach is described in [Schedl, 2008]. For each artist A_i, the top-ranked Web pages returned by a search engine are retrieved and searched for occurrences of all other artist names A_j in the collection. The number of page hits represents a co-occurrence count, which equals the document frequency of the term "A_j" in A_i's pages. In relation to the number of A_i's total pages, this gives an (non-symmetric) estimate of similarity. For a symmetrised measure, A_i's similarity to A_j and A_j's similarity to A_i are averaged. A similar technique can also be applied to micro-blogging posts ("tweets") retrieved from *Twitter*[10] [Schedl, 2010].

Other approaches do not directly search for co-occurrences on pages. Instead, they rely on the page counts returned for search engine requests. Assessment of similarity from search engine page counts can be considered an abstraction of the before mentioned approach to co-occurrence analysis. As opposed to [Cohen and Fan, 2000] and [Schedl, 2008], the severe shortcoming of page-count-based approaches is that in order to create a full similarity matrix, the number of necessary search engine requests is quadratic in the number of artists. Therefore, these approaches scale poorly to real-world music collections.

[Schedl et al., 2005a] define similarity as the conditional probability that A_i appears on a Web page known to mention A_j. Since page counts for the queries *"A_i"* and *"A_i"+"A_j"* indicate the relative frequency of this event, the conditional probability can be estimated. Also for this scenario, it is necessary to restrict search results to Web pages presumably relevant to music (e.g., by adding the keywords *music review*), since unconstrained queries would lead to unjustifiably higher page counts for common speech artist names (e.g., "Kiss") and therefore distort the similarity relations (cf. Section 2.1.2.3). The similarity measure proposed by [Schedl et al., 2005a] is also initially non-symmetric and should be averaged if a symmetric measure is required. Alternatively, the non-symmetric similarity matrix can be exploited to uncover prototypical artists [Schedl et al., 2005b, Schedl et al., 2005c]. In general, page counts are a flexible tool for estimating relatedness of concepts. For instance, [Schedl et al., 2006b] assess artist-genre relations using page counts. Another application scenario is (country-specific) artist popularity estimation [Schedl et al., 2010a]. Further explored sources for this task are micro-blogging posts, Last.fm usage statistics, and shared folders on P2P networks (cf. Section 2.1.2.8).

[Zadel and Fujinaga, 2004] restrict similarity calculation to potentially related artists by invoking the *Amazon.com* service *Listmania!*[11] prior to using Google. Listmania! provides user-compiled lists of related items and can henceforth be a

[8] http://www.altavista.com

[9] was: http://www.northernlight.com, now: http://www.nlsearch.com

[10] http://twitter.com

[11] http://www.amazon.com/gp/help/customer/display.html?nodeId=14279651

valuable source of cultural similarity. However, it has to be kept in mind that long-tail-effects such as a popularity bias may affect this pre-filtering step.

2.1.2.7 Playlists

Another source of co-occurrences of music is playlists. The idea is that songs that are similar in a sense, are likely positioned closely within playlists since smooth transitions are usually intended. One of the earlier approaches can be found in [Pachet et al., 2001], where airplay lists from a French radio station and compilation CD track listings from *CDDB*[12] are examined to extract co-occurrences between tracks and between artists.

[Cano and Koppenberger, 2004] create a similarity network via extracting playlist co-occurrences of more than 48,000 artists from a total of more than 29,000 playlists retrieved from *Art of the Mix*[13] — a Web service that allows users to upload and share their mix tapes or playlists. In the network, a connection between two artists is made if they co-occur in a playlist. The paper reveals some interesting properties. For instance, one large cluster of nodes connects more than 99% of the artists but each artist itself is only connected with a small number of other artists. Hence, such a similarity measure can only capture (strong) positive similarities between artists. The probability of indirect links, i.e., that two neighbours of a given artist are also connected, is low.

In a more recent paper that exploits playlists to derive similarity information, [Baccigalupo et al., 2008] analyse co-occurrences of artists in more than 1 million playlists publicly shared by the MusicStrands community. The authors extract the 4,000 most popular artists from the full playlist set, measuring the popularity as the number of playlists in which each artist occurs. Further, by introducing a distance parameter, it is taken into account that two artists consecutively occurring in a playlist are probably more similar than two artists occurring farther away. From their measure, the authors calculate for each artist a "genre affinity" value to 26 different genres, as well as artist-to-artist affinities. Additionally, normalisation is performed to account for the popularity bias, i.e., very popular artists co-occur with a lot of other artists in many playlists.

2.1.2.8 Peer-to-Peer Network Co-Occurrences

P2P network users are commonly willing to reveal various kinds of meta-data about their shared content. In the case of shared music files, file names and ID3 tags are disclosed and represent a valuable source of usage data and co-occurrences in real-world collections.

In early work, data extracted from the P2P network *OpenNap*[14] is used to derive music similarity information [Whitman and Lawrence, 2002, Ellis et al., 2002, Logan et al., 2003, Berenzweig et al., 2003]. [Logan et al., 2003] and [Berenzweig et al., 2003]

[12] CDDB is a Web-based CD identification service that returns, for a given disc identifier, meta-data like artist and album name, track listing, or release year. This service is offered in a commercial version operated by *Gracenote* (`http://www.gracenote.com`) as well as in an open source implementation named *freeDB* (`http://www.freedb.org`).

[13] `http://www.artofthemix.org`

[14] `http://opennap.sourceforge.net`

further determine the 400 most popular artists on OpenNap (by the time of 2002) and gather meta-data on shared content, resulting in 175,000 user-to-artist relations from about 3,200 shared music collections. A finer analysis of usage data to estimate the popularity of artists in specific countries is proposed in [Schedl et al., 2010a]

Additionally, [Logan et al., 2003] compare similarities defined by P2P co-occurrences, AMG expert reviews, playlist co-occurrences, data gathered from a survey, and content-based similarity. To this end, they compare similarity rankings according to each data source. The main findings for P2P data are a high sparsity, a high degree of overlap with playlist data (cf. Section 2.1.2.7), and a low agreement with the content-based measure (which was also the case for all other sources except for AMG reviews).

A recent approach that derives similarity information on the artist and on the song level from the *Gnutella* P2P file sharing network[15] is presented by [Shavitt and Weinsberg, 2009]. The authors collect meta-data of shared files from more than 1.2 million Gnutella users, yielding a data set of 530,000 songs. One finding of analysing the resulting data is that users' shared collections have large overlaps in terms of contained songs. Therefore, also a subset of the data seems to be representative, making exhaustive crawls unnecessary. The data gathered is used for artist recommendation and for song clustering.

[Anglade et al., 2007] cluster users of a P2P network based on their music preferences to automatically create virtual communities. User-to-user similarity is derived by applying methods typically used for collaborative filtering.

2.1.2.9 Collaborative Filtering-based Approaches

Another source from which to derive contextual information is exploiting users' listening habits, their shopping behaviour, as well as explicit user feedback. This approach is also known as *collaborative filtering* (CF). To perform this type of similarity estimation typically applied in recommender systems, one must have access to a (large and active) community. Since the relevant CF systems are commercial applications (e.g., Last.fm or Amazon[16]), implementation details are usually concealed. In general, two types of similarity relations can be inferred by tracking users' habits: item-to-item similarity (where an item could potentially be a track, an artist, a book, etc.) and user-to-user similarity. For example, when representing preferences in a user-item matrix S, where $S_{i,j} > 0$ indicates that user j likes item i (e.g., j has listened to artist i at least once or j has bought product i), $S_{i,j} < 0$ that j dislikes i (e.g., j has skipped track i while listening or j has rated product i negatively), and $S_{i,j} = 0$ that there is no information available (or neutral opinion), user-to-user similarity can be calculated by comparing the corresponding M-dimensional column vectors (where M is the total number of items), whereas item-to-item similarity can be obtained by comparing the respective N-dimensional row vectors (where N is the total number of users) [Linden et al., 2003, Sarwar et al., 2001]. For a detailed discussion of CF for music recommendation in the long-tail and real-world examples from the music domain, the reader is referred to [Celma, 2008].

[15] http://rfc-gnutella.sourceforge.net

[16] http://www.amazon.com

2.1.2.10 Other Sources of Context-Based Information

While the preceding sections cover the currently best established and most prominent sources to derive context-based music information, there also exist alternative data sources that are considered in the literature, for instance, as previously mentioned, *server logs*. [Cohen and Fan, 2000] analyse download statistics from an *AT&T*-internal digital music repository which was monitored for three months, yielding a total of nearly 30,000 music downloads relating to about 1,000 different artists. Conceptionally, this technique is very similar to exploiting P2P network co-occurrences (cf. Section 2.1.2.8).

Another approach is the analysis of playlist *user ratings* [Slaney and White, 2007]. Using the Yahoo! music service[17], music piece similarity is obtained by comparing normalised rating vectors. [Jacobson et al., 2008] propose the usage of *social network data* from *MySpace*[18]. From the inherent artist network, "communities" of similar artists are detected that exhibit structures closely related to musical genre. [Fields et al., 2008] further propose to combine artist relations extracted from MySpace with content-based similarity for automatic playlist generation (cf. Section 2.1.3).

Music-related data also includes images such as band photographs or album artwork. [Schedl et al., 2006a] present methods to automatically retrieve the correct album cover for a record. [Brochu et al., 2003] use colour histogram representations of album covers to index music pieces. [Lībeks and Turnbull, 2010] calculate music similarity based on artists' promotional photographs. It is shown that the notions of similarity and genre to some extent correspond to visual appearance.

2.1.3 Hybrid Approaches

As content-based and context-based features stem from different sources and represent different aspects of music, they can be beneficially combined in order to outperform approaches based on just one source or to mutually compensate for limitations. For example, to accelerate the creation and improve the quality of playlists, in [Knees et al., 2006a] and [Pohle et al., 2007a], Web-term-based artist profiles are pre-clustered using a Self-Organizing Map (SOM, [Kohonen, 2001], cf. Section 3.3.1) to find promising relations between artists. Based on the neighbourhood on the SOM, audio similarities between tracks of similar artists are calculated. Not only does this filtering step reduce the number of necessary audio similarity calculations, it also reduces the number of (audio-based) outliers in the resulting playlists. [Schnitzer et al., 2007] make use of Last.fm tags instead of Web-based profiles for pre-clustering and playlist labelling. [Fields et al., 2008] extract related artists from MySpace and incorporate a content-based approach to measure similarity of tracks by these artists for automatic playlist generation.

[Donaldson, 2007] proposes a hybrid recommender built upon collaborative filtering data (item similarity) and acoustic features that is capable of disambiguating the user's music-seeking intentions. [Shao et al., 2009] combine content-based features with user access patterns to improve music recommendation.

[17] `http://music.yahoo.com`
[18] `http://www.myspace.com`

[Whitman and Smaragdis, 2002] use audio-based and Web-based genre classification for the task of style detection on a set of 5 genres with 5 artists each. Combining the predictions made by both methods linearly, yields perfect overall prediction for all test cases. [McKay and Fujinaga, 2008] examine combinations of signal-based, symbolic, and cultural features for genre classification. [Baumann, 2005] linearly combines audio-based track similarity with Web-based artist similarity to obtain a new similarity measure. [Whitman, 2005] uses audio features and semantic descriptors to learn the meaning of certain acoustic properties and to overcome the so-called "semantic gap". Similarly, to improve the quality of classification according to certain categories like genre, instrument, mood, or listening situation, [Aucouturier et al., 2007] combine timbre similarity with contextual meta-data. [Barrington et al., 2009] present a combined method for retrieval that incorporates multiple sources of features (i.e., acoustic features related to timbre and harmony, social tags, and Web documents). These largely complementary information sources are combined to improve prediction accuracy of a set of semantic categories (cf. Section 2.3).

2.2 Web Information Retrieval and Search Engines

Until the 1990s, efficient organisation and indexing of text documents to facilitate access to the contained information and/or the documents themselves was mainly the concern of librarians and experts. Since then, with the advent of the World Wide Web, interest in IR technologies has dramatically increased and today, IR technologies are an every day commodity.

IR "deals with the representation, storage, organization of, and access to information items" and comprises tasks such as "modeling, document classification and categorization, systems architecture, user interfaces, data visualization, filtering, languages, etc." [Baeza-Yates and Ribeiro-Neto, 1999], p.1–2. The central goal of IR is to satisfy the information need of the user. From the user's perspective, in current systems, this need (unfortunately) has to be expressed directly in a machine understandable form, i.e., a query usually consisting of a sequence of natural language keywords. The IR system, on the other side, is required to have a representation of indexed documents that models the contained information in some fashion to be able to match the information need.

To this end, documents are typically modelled using a set of index terms or keywords directly extracted from the contents of the documents. Commonly, a weight is also assigned to indicate the importance of each term for the respective document. The given query is processed and compared with the indexed documents to retrieve documents which are then ranked by relevance (cf. Section 3.1.3). Relevance is predicted according to the underlying information model. Upon presentation, the user may start to examine this ranking (i.e., the search result) for the desired information. Optionally, the user may be given the possibility to indicate relevant results and to start a user feedback circle in which the query formulation is iteratively modified to improve search results (cf. [Baeza-Yates and Ribeiro-Neto, 1999], p.9).

Due to the rapid growth of the Web — especially in the last decade — it is now the world's largest publicly accessible repository of data. Without knowing how many Web pages really exist (estimates of the total number of Web documents range from 231.5 million Websites[19] to at least 25 billion pages[20]), it is clear that there is an ever growing demand for technologies that assist the user in finding relevant information within this plethora. Moreover, it seems that only the development of accurate search engines like Google has made the Web accessible and attractive to use to a wide audience, and thus laid the foundations to pervade the mainstream society. Apart from the sheer number of documents which need to be indexed, the Web poses additional challenges to IR systems.

2.2.1 Web Data Mining

Data mining is generally concerned with the discovery of patterns and knowledge in various data sources. Web data mining, as a more specific task, "aims to discover useful information or knowledge from the Web hyperlink *structure*, page *content*, and *usage* data" [Liu, 2007], p.6. For tailoring methods to exploit these three sources, some unique characteristics of Web data (that distinguish it from traditional database mining and text mining) have to be taken into account (cf. [Liu, 2007], p.4-5):

- The Web contains a huge amount of most diverse information that covers virtually any topic.
- The Web is highly heterogeneous in terms of contained data types (i.e., all types of multimedia data), structuring of data (e.g., tables vs. unstructured text), and information (e.g., different formulations of redundant information).
- Information is typically linked via hyperlinks.
- Web data is noisy, first, due to heterogeneous content that often includes navigation links or advertisements, and second, because contents on the Web are arbitrary, i.e., information is often low in quality, contains errors, or is (intentionally) misleading. Especially given that the Web has developed into an enormous economic factor, some issues that have to be taken into consideration are spam and other techniques which aim at exploiting the nature of information ranking algorithms to acquire more attention — and in consequence more revenue.
- The Web is used commercially and for (public) services.
- Information on the Web is dynamic, i.e., the Web is subject to constant change. However, outdated information may persist on the Web and impose difficulties to data mining approaches if not contextualised chronologically.
- The Web is a social medium, i.e., people communicate via forums, blogs, or comments, and participate in activities ("Web 2.0").

In the remainder of this section, the focus is put on methods that mine information useful for indexing and retrieval.

[19] Source: *WolframAlpha*, URL: `http://www.wolframalpha.com/input/?i=number+of+websites` (estimation as of April 2009, page accessed 29-Jun-2010)

[20] `http://www.worldwidewebsize.com` (accessed 30-Jun-2010)

To extract information from Web data, the first step consists in obtaining the data from the Web. This is usually accomplished automatically by using a Web crawler. Starting from a seed URL, a crawler follows the contained hyperlinks recursively to find other Web resources. Crawlers that collect as many pages as possible with the aim of indexing the complete Web (as used by Web search engines) are called universal crawlers, as opposed to focused crawlers that try to download and index only Web pages of a certain type (cf. [Chakrabarti et al., 1999]).

After collecting data, the mere application of traditional, purely content-based text-IR methods to Web documents is not sufficient. First, the number of available documents on the Web is too large, i.e., it is difficult to select a small subset of the most relevant Web pages from millions of pages that all contain the desired keywords. Second, content-based relevance ranking methods are prone to spamming due to the nature of the used algorithms (cf. Section 3.1). Thus, for indexing the downloaded content, the link structure, i.e., relations between the Web pages, (as well as — in later steps — the usage data[21]) gives important additional information to determine the "authority" and thus the relevance of a page. Two examples of ranking algorithms that incorporate information on hyperlink structure are *HITS* (see [Kleinberg, 1999]) and *PageRank* (see [Page et al., 1999]), which is used within the Google ranking scheme.

HITS (short for *Hypertext Induced Topic Search*) starts from a ranking for a given query and expands the result set by including pages that either point to any page in the result set or that are pointed to from any page in the set. On that expanded set, an authority ranking and a hub ranking are produced by computing eigenvectors. An authority page is defined as having many in-links, i.e., hyperlinks on many other pages point to it. A hub page is defined as having many out-links. The basic idea of HITS "is that a good hub points to many good authorities and a good authority is pointed to by many good hubs. Thus, authorities and hubs have a mutual reinforcement relationship." [Liu, 2007], p.255. The ability of HITS to rank pages with respect to a query is considered to be a main strength. However, HITS can be easily influenced by adding out-links to highly ranked authority pages, making it susceptible to spam pages. Furthermore, on-line expansion of the result set and eigenvector calculation at query time are both time consuming tasks.

In contrast to HITS, in the PageRank algorithm, the "prestige" of a Web page is static within a given set of Web pages, i.e., PageRank for a page is defined query-independently and can thus be calculated off-line. PageRank favours pages that have many in-links. More precisely, PageRank of a page i is basically defined as the sum of PageRank values of all other pages pointing to i. This recursive definition leads to a system of n linear equations with n unknowns, where n is the number of Web pages. For calculation, an iterative process that approximates the principal eigenvector of the adjacency matrix of all Web pages is applied. By prioritising in-links, compared to HITS, PageRank is less susceptible to spamming. Together with the efficient off-line calculation that allows for fast ranking at query time, this is considered to be a contributing factor to Google's immense success (cf. [Liu, 2007], p.254).

[21]While in principle it is possible for anybody to build a Web search engine from scratch by designing crawlers and analysing Web content and link structure, to exploit the third source of Web data mining, i.e., usage data, one must already have a running system frequented by people, since this sort of data becomes only available from monitoring users' behaviour.

2.2.2 Multimedia Indexing

The IR methods discussed so far all deal with retrieval of texts. However, the Web contains all types of media that also has to be organised and made accessible in some way. For indexing databases of multimedia files, typically predefined categories of meta-data are used. Furthermore, for images, videos, and sound, it is possible to enable query-by-example search, i.e., the "query" is itself an image (or a sound, respectively, cf. Section 2.3.2) and by extracting features from the query, similarities to the items in the database (rather, their feature representations) can be calculated and a relevance ranking is obtained (cf., for instance, [Lew et al., 2006, Smeaton et al., 2008]).

On the Web, both these types of retrieval are rather impractical. First, explicit meta-data generation is infeasible due to the sheer amount of data already on the Web — not to mention the huge amounts of new content added everyday. Community-based approaches for tagging are an attempt to deal with this (as are tagging games, cf. Section 2.1.2.2), but, despite all efficiency of the distributed workload, they are unlikely ever to be capable of handling and annotating the amounts of data presented. The idea of using query-by-example methods to find multimedia data, on the other hand, is in principle applicable to the Web. Examples of such services are *TinEye*[22] for image search and *Owl Music Search*[23] for music (see below). These services, nevertheless, make the upload of an example media file necessary, which requires preparation, consumes bandwidth, and takes time. This is likely one reason why these systems are lacking the same acceptance as established Web search engines. However, in existing Web image retrieval engines, an option to find images similar to search results is frequently included. Thus, the query file is already present and features are pre-calculated, which makes this approach more applicable.

To bring image search to the Web scale and allow instant presentation of relevant results, images are typically indexed using their Web context. More precisely, text data that can be associated to the image, such as the filename, anchor texts that link to the image, or the text adjacent to the image on embedding Web pages is used to automatically describe the content (e.g., [Frankel et al., 1996, Tsymbalenko and Munson, 2001, Kherfi et al., 2004]). Although the underlying techniques are not published exhaustively, it can be assumed that the image search services of state-of-the-art Web search providers[24,25,26] follow very similar approaches. For videos and music, similar techniques are conceivable. Currently, the most prominent strategy seems to rely on meta-data like title, tags, and an optional text description often available on portals dedicated to enabling anyone to share multimedia content such as *YouTube*[27].

[22] `http://www.tineye.com`
[23] `http://owlmm.com`
[24] `http://images.google.com`
[25] `http://images.search.yahoo.com`
[26] `http://www.bing.com/images`
[27] `http://www.youtube.com`

2.3 Music Search Engines

This section deals with approaches to retrieve music from databases or from the Web. In contrast to image search, the idea of using contextual Web data to index music files found on the Web has not found its way into many (commercial) applications. The reason for this does not necessarily need to be of technical nature or due to inapplicability to this task, but could also be linked to the fact that providers of such a search functionality aim at avoiding any legal grey area, since many of the audio files found on the Web potentially infringe on a copyright.

However, there exist some search engines that use specialised (focused) crawlers to find all types of sounds on the Web. As with Web image search, the traced audio files are then indexed using contextual information extracted from the text surrounding the links to the files. Examples of such search engines are *Aroooga* by [Knopke, 2004] and *FindSounds*[28]. Note that the music indexing approach presented in Chapter 5 also makes use of contextual text information from the Web. Compared to the above mentioned approaches a direct Web context is not available since the indexed music pieces do not necessarily originate from the Web. Hence, a textual context has to be constructed artificially by finding Web pages that mention the meta-data of tracks.

In the following sections, further approaches to music retrieval are reviewed. First, exemplary retrieval methods that build upon symbolic representations of music are presented. Second, approaches that use an actual piece (or snippet) of music as query are discussed. Systems that allow for cross-media retrieval, more precisely, query-by-text music retrieval systems, are reviewed third.

2.3.1 Symbolic-Representation-Based Retrieval

Most of the methods to retrieve music from databases proposed so far operate on symbolic representations (frequently derived from MIDI notations). The *Themefinder* Web search engine[29], for instance, allows for querying a symbolic database by entering melodic sequences in specific text formats, cf. [Kornstädt, 1998].

Other systems that follow more intuitive approaches are also usable for less musically educated users. For example, in query-by-humming/singing systems the user can hum or sing a part of the searched piece into a microphone. From that recording, musical parameters (typically related to melody) are extracted and the obtained sequence serves as a query to the database (cf. [Ghias et al., 1995]).

An example of a search engine offering exhaustive possibilities for querying is *Musipedia*[30]. Musipedia indexes a large number of music pieces by crawling the Web for MIDI files that can then be used for identification of pieces. For indexing of pieces, the melodic contour, pitches and onset times, and a rhythm representation are extracted. To find a piece in the database, a theme (i.e., the query) can be either entered in Parsons code notation [Wikipedia, 2010e] or whistled into a microphone (to find matching melodies), played on a virtual piano keyboard (to find matching pitch and onset sequences), or tapped on the computer keyboard (to find matching

[28] `http://www.findsounds.com`
[29] `http://www.themefinder.org`
[30] `http://www.musipedia.org`

rhythms). For a detailed explanation of the incorporated techniques, as well as a comprehensive comparison of symbolic music retrieval systems and MIR systems in general, see [Typke, 2007].

2.3.2 Audio-Based Retrieval

All audio-based similarity approaches that find the most similar tracks in a collection for a given track perform query-by-example retrieval. Therefore, an audio-based search engine is implicitly included in many systems that offer exploration of the music space by recommending similar sounding pieces. An example of such a service is the *FM4 Soundpark* music player[31] that recommends other songs based purely on the acoustic content (see [Gasser and Flexer, 2009]). The FM4 Soundpark is a moderated open platform for up-and-coming artists hosted by the Austrian public broadcasting station FM4 and targets primarily alternative music. Especially in this case, where artists are generally unknown, content-based similarity is the method of choice for recommendation.

Another system that incorporates this capability, but also explicitly offers querying of the database by uploading an audio file, is the Owl Music Search engine already mentioned in Section 2.2.2. This service allows searching for music from commercial and independent labels, with a focus on music published under a *Creative Commons* license[32]. To this end, the song catalogues of music sites such as *ccMixter*[33], *Magnatune*[34], and *Jamendo*[35] are indexed. [Maddage et al., 2006] propose a vector space retrieval method for music. In their approach, music is modelled in a hierarchical scheme incorporating beat structure, harmony events (that model chord information), and acoustic events (defined as the occurrence of instrumental and vocal content).

Special cases of query-by-example systems are fingerprinting services such as the commercial service *Shazam*[36]. Instead of retrieving similar music pieces, the aim of these systems is to identify a query song (more precisely, a specific recording) and to return the associated meta-data (artist name, title, etc.). Typically, the query consists of a short, low quality recorded portion of a music piece obtained via a cellular phone. Hence, the feature extraction process (that generates the fingerprints) must be robust against all kinds of distortions caused by factors such as cheap microphones, cellular phone connections, or background noises during recording time to create a unique descriptor that can be matched to the indexed fingerprints in the collection, cf. [Wang, 2006].

2.3.3 Text-Based Retrieval

An even more challenging task is to design systems that enable cross-media retrieval. In this case, systems that allow queries consisting of arbitrary natural language text (e.g., descriptions of sound, mood, or cultural events) and that return music pieces

[31]http://fm4.orf.at/soundpark
[32]http://creativecommons.org
[33]http://ccmixter.org
[34]http://www.magnatune.com
[35]http://www.jamendo.com
[36]http://www.shazam.com

semantically related to this query, are of interest. Apart from the two text-based systems mentioned above, the number of systems offering such query modalities is rather small. The most elaborate approach so far has been presented by [Baumann et al., 2002]. Their system is supported by a semantic ontology which integrates information about the artist, genre, year, lyrics, and automatically extracted acoustic properties like loudness, tempo, and instrumentation, and defines relations between these concepts. Besides the correct extraction of the features, the mapping of the query to the concepts in the ontology has to be accomplished. In the end, the system allows for semantic queries like *"something fast from ..."* or *"something new from ..."*. Also phonetic misspellings are corrected automatically.

[Celma et al., 2006] present the music search engine *Search Sounds*[37]. The system uses a special crawler that focuses on a set of manually defined "audio blogs", which can be accessed via RSS feeds. In these blogs, the authors explain and describe music pieces and make them available for download (whether legally or illegally depends on the blog). Hence, the available textual information that refers to the music, together with the meta-data of the files, can be used to match text queries to actual music pieces. Furthermore, for all returned pieces, acoustically similar pieces can be discovered by means of content-based audio similarity. Audio similarity is also exploited by [Sordo et al., 2007] for automatically propagating text labels.

Another system that enhances music search with additional semantic information is *Squiggle* by [Celino et al., 2006]. In this framework, queries are matched against meta-data and further evaluated by a word sense disambiguation component that proposes related queries. For example, a query for *"rhcp"* results in zero hits, but suggests to search for the band *Red Hot Chili Peppers*; searching for *"Rock DJ"* proposes the song by *Robbie Williams*, the genre Rock, as well as other artists (all DJs). The underlying semantic relations are taken from the freely available community databases *MusicMoz*[38] and *MusicBrainz*[39]. However, although semantic relations are integrated, the system depends on explicit knowledge which is in fact a more extensive set of manually annotated meta-data.

A system that can be used for text-based music search and that is not restricted to a pre-defined set of meta-data is the collaborative music platform Last.fm. Besides keeping track of user's listening habits for recommendation, it enables users to assign tags to the tracks in their collection (cf. Section 2.1.2.2). These tags provide a valuable source of information on how people perceive and describe music and can be used for indexing of the music repository. However, tagging is often inconsistent and noisy. In the present thesis, a (stable) subset of Last.fm data serves as a ground truth for evaluating some of the proposed methods.

Several approaches aim at indexing music with arbitrary vocabularies, i.e., automatically assigning music-related descriptors — sometimes also referred to as "semantic labelling" — by building models that incorporate acoustic properties from low-level features. In early work, [Whitman, 2005] maps low-level characteristics of audio signals to semantic concepts to learn the "meaning" of certain acoustic properties. [Turnbull et al., 2007a] present a method for semantic retrieval that is based on the CAL500 data set (see Section 5.6.1.2). Models of these properties

[37] `http://www.searchsounds.net`
[38] `http://www.musicmoz.org`
[39] `http://www.musicbrainz.org`

are learned from audio features and can be used to label previously unseen tracks. Correspondingly, the system can also be used to search for relevant songs based on queries consisting of the words used for annotation. [Barrington et al., 2009] further extend this approach by incorporating multiple (and largely complementary) sources of features (i.e., acoustic features related to timbre and harmony, social tags, and Web documents). These information sources are combined via machine learning methods to improve prediction accuracy. Usefulness of combining contextual data with content-based features is theoretically founded and also demonstrated by [Aucouturier et al., 2007], who improve classification according to certain meta-data categories like genre, instrument, mood, or listening situation by exploiting correlations between meta-data categories.

2.4 User Interfaces to Music Collections

This section reviews different user interfaces to music collections. In general, such systems use information on music to automatically structure a repository and aid the user in exploring the contents. A short discussion of Music Information Systems is given, followed by an overview of the large number of available map-based interfaces. Finally, other intelligent music interfaces are presented.

2.4.1 Music Information Systems

A classic way for accessing music collections or information on music is via a (typically Web-based) information system. Examples of music information systems are allmusic, Yahoo! Music, Last.fm, and also (if restricted to the music domain) *Wikipedia*[40]. Informally defined, a music information system serves as a kind of multimedia encyclopedia that has entries for different musical entities on different levels of granularity as well as links between these entity descriptions. In practice, typical categories included in such a system are discographies, biographical information, and line-up for bands, as well as track listings, cover artwork, and other meta-data for albums. Furthermore, recommendations such as similar artists or albums are included. The presented data may either originate from editors (as is the case for allmusic and Yahoo! Music, for instance) or from a community (as in the case of Last.fm and Wikipedia). A detailed discussion on music information systems, as well as an approach towards the automatic generation of such a system from Web data can be found in [Schedl, 2008].

2.4.2 Map-based Interfaces

The idea of map-based music interfaces is to organise musical entities in a two-dimensional layout to display the global composition of collections and intuitively visualise similarity by relating it to closeness on the map. Furthermore, orientation on a map is a concept familiar to people and therefore a particularly good choice to be incorporated into interfaces for novel purposes. This kind of structuring allows then for browsing music collections by examining different regions on the map as well as for implicit recommendation by exploring areas surrounding known pieces.

[40] `http://www.wikipedia.org`

Most systems that create a map for music organisation rely on a *Self-Organizing Map (SOM)*. In seminal work, [Rauber and Frühwirth, 2001] propose this approach to form clusters of similar sounding songs and to project them on a two-dimensional plane (cf. Section 3.3). [Pampalk, 2001] extends this approach to create the *Islands of Music* interface. For the Islands of Music, a SOM is calculated on *Fluctuation Pattern* features and visualised by applying a technique called *Smoothed Data Histogram*. Finally, a colour model inspired by geographical maps is applied (see Section 3.3.2). In addition, several extensions have been proposed, e.g., the usage of Aligned SOMs to enable a seamless shift of focus between different aspects of similarity (cf. [Pampalk et al., 2004]) or a hierarchical component to cope with large music collections (cf. [Schedl, 2003]). [Neumayer et al., 2005] utilise SOMs for browsing in collections and intuitive playlist generation on portable devices. [Leitich and Topf, 2007] propose mapping to a sphere instead of a plane by means of a GeoSOM to create the *Globe of Music*. Other music map approaches use SOM derivatives (e.g., [Mörchen et al., 2005]) or a k-nearest neighbor graph (as in [Seyerlehner, 2006]). [Vembu and Baumann, 2004] use textual information from Amazon reviews to structure music collections via a SOM by incorporating a fixed list of musically related terms to describe similar artists.

For deepening the impression of exploration of music collections, also three-dimensional extensions to music maps are available. Hence, the user can move through a virtual terrain to find new music. The first approach in this direction can be found in [Knees et al., 2006b] (for a detailed description see Section 4.3.1). Recent alternatives can be found in [Lübbers and Jarke, 2009] or [Gasser and Flexer, 2009].

Apart from generating music maps that resemble geographical maps, other approaches to visualise the music space in two dimensions have been presented. Frequently, these interfaces utilise *multidimensional scaling (MDS)* data projection techniques (see, e.g., [Cox and Cox, 2000]). For example, [Cano et al., 2002] incorporate a FastMap in the visualisation of the *SongSurfer* interface. In the *MusicGalaxy* interface by [Stober and Nürnberger, 2010], a pivot-based MDS is applied. MusicGalaxy combines multiple sources of similarity information, giving the user control over their influence. Furthermore, when exploring the map, a magnification of similar tracks supports browsing by compensating for data projection deficiencies. In the *Search Inside the Music* interface, [Lamere and Eck, 2007] utilise an MDS approach for similarity projection into a three-dimensional space. A similarity-graph-based interface for portable devices is presented by [van Gulik et al., 2004]. [Lillie, 2008] applies a *principal components analysis (PCA)* to project multidimensional music descriptors to a plane in the *MusicBox* framework. Furthermore, [Lillie, 2008] gives another comprehensive overview of user interfaces to music collections.

Other map-based interfaces enable additional interaction methods by exploiting specific devices, foremost tabletop displays. [Hitchner et al., 2007] use a tabletop display to facilitate browsing and rearrangement of a music collection structured via a SOM. A similar approach that makes use of a *Bricktable* multi-touch interface for structuring Electronica music for DJs is presented by [Diakopoulos et al., 2009]. The *SongExplorer* by [Julià and Jordà, 2009] structures large collections and allows for interaction using a *reacTable*-like interface. [Baur et al., 2010] exploit lyrics-based descriptors to organise collections in the *SongWords* tabletop application. The *MUSICtable* interface presented by [Stavness et al., 2005] can utilise different approaches to map creation (including manual construction) and is intended to enforce social interaction when creating playlists.

Finally, for map-based interfaces, the underlying maps do not have to be necessarily created artificially. For instance, music collections (or music-related data) may be made available by distributing them over real geographical maps. [Celma and Nunes, 2008] use data from Wikipedia for placing bands on their corresponding position on a world map. [Govaerts and Duval, 2009] extract geographical information from biographies on the Web and utilise this information, e.g., to visualise radio station playlists.

2.4.3 Other Intelligent Interfaces

Although there exist many interesting approaches that are based on manually assigned meta-data (e.g., [Torrens et al., 2004] or *musiclens*[41]), this section will primarily deal with systems that rely on automatic feature calculations of music pieces or artists and are therefore capable of "intelligently" structuring and presenting music collections automatically.

[Pampalk et al., 2005] use hierarchical one-dimensional SOMs on Web-based artist term profiles to guide the user to relevant artists. At each level, the user chooses from sets of music descriptions that are determined via term selection approaches (cf. sections 2.1.2.3 and 4.2). [Pohle et al., 2007b] performs an NMF on Web term profiles for artists which yields a weighted affinity of each artist to each resulting "semantic" concept. In the resulting *Artist Browser*, the influence of each concept can be adjusted manually to find best fitting artists and to display related Web content. [Schedl and Pohle, 2010] propose the *Three-Dimensional Co-Occurrence Browser* to browse collections of Web pages for multimedia content by selecting descriptive terms.

A very remarkable interface to discover new pieces and easily generate playlists is presented by [Goto and Goto, 2005]. From streams of music pieces (represented as discs), the user can simply pick out a piece to listen to or "collect" similar pieces by dragging a seed song into one of the streams. The different streams describe different moods. The number of released discs can be regulated for each mood separately by "tabs". Furthermore, the system invites users to experiment with playlists as all modifications can be undone easily by a so called time-machine function. The intuitive drag-and-drop interface also facilitates the combinination of playlists.

The *Traveller's Sound Player* is an interface for accessing music on mobile music players by computing song similarities from extracted audio features and creates a placement for all songs on a circular path using a *travelling salesman problem (TSP)* algorithm, cf. [Pohle et al., 2007a]. This arrangement permits the user to quickly locate music of a particular style by simply turning a wheel, much like searching for radio stations on a radio. [Schedl et al., 2006b] augment this interface with genre descriptors extracted via co-occurrence analysis. Following this line of research, [Schnitzer et al., 2007] make use of Last.fm tags for pre-clustering and playlist labelling. In the resulting *Intelligent iPod* prototype, the click wheel on Apple's iPod can be used to navigate linearly through an entire music collection, automatically arranged according to musical similarity.

[41] `http://finetunes.musiclens.de`

Another extension of the Traveller's Sound Player that is not aiming at portable music players is the *MusicRainbow* by [Pampalk and Goto, 2006]. In this interface, artist similarity is computed from extracted audio features and — using a TSP algorithm — artists are placed on a "circular rainbow", where coloured arcs reflect the genre of each artist. Furthermore, the interface is enriched with descriptive terms from artist-related Web pages. To give the user more choice in selecting relevant dimensions of similarity, the subsequentially developed *MusicSun* interface combines three types of similarity, namely audio-based similarity, Web-based similarity, and word-based similarity, cf. [Pampalk and Goto, 2007]. Word-based similarity allows to focus on specific words instead of overall term similarity and supports the user in finding other artists that are strongly connected to these words.

Chapter 3

Methodological Background

In this chapter, the methods underlying the techniques proposed in the subsequent chapters are reviewed. The methods are structured into three main areas: Web Mining and Document Indexing (Section 3.1), Audio Similarity (Section 3.2), and Music Maps (Section 3.3). Note that Section 3.1 — apart from Web information retrieval methods — also reviews methods originally not belonging to Web mining and document indexing, i.e., supervised learning and classification (here used for Web page classification) and multiple sequence alignment (here used for redundancy detection in Web pages and noise removal).

3.1 Web Mining and Document Indexing

This section deals with methods that are used in the next chapters in the context of Web retrieval and processing of Web content. The first three sections deal with standard Web retrieval and indexing techniques, namely fetching and processing of Web data (Section 3.1.1), storing, compressing, and accessing the fetched data (Section 3.1.2), and assessing relevance of stored Web documents with respect to a given query (Section 3.1.3). Since for the practical realisation of the proposed methods, these three steps are performed using the Java-based open source Web crawler *Nutch*[1] and the underlying open source indexer *Lucene*[2], the indexing and retrieval variants as implemented in these packages are emphasised. Section 3.1.4 deals with supervised learning and reviews the random forest classifier used for page classification. Section 3.1.5 reviews a technique for multiple sequence alignment which is here applied for identifying redundant parts on Web pages.

3.1.1 Web Data Retrieval

Since the primary tasks addressed in the following are Web indexing and retrieval, the general design of a Web search engine (as far as needed for understanding the proposed methods) is presented. In Web Retrieval, the first step consists in obtaining data from the Web to operate on. To this end, typically a Web crawler is used to discover and download millions of Web pages. The principal idea of a Web crawler, i.e., recursively following the hyperlinks on Web pages to find other Web resources

[1] `http://lucene.apache.org/nutch/`
[2] `http://lucene.apache.org/`

starting from one or more seed URLs, has already been discussed in Section 2.2.1. Here, a more detailed description is given (cf. in the following [Liu, 2007], p.274*ff* and p.286*ff*).

A central component of a crawler is the so called *frontier*. The frontier is a list of unvisited URLs that is initialised with the seed URLs and constantly updated while new URLs are extracted from the crawled Web pages. In case the frontier is implemented in a first-in-first-out manner, i.e., the sequence of pages to be visited next depends on the sequence of pages visited without any further reordering, the crawler is said to be a *breadth-first* crawler. In case the frontier is implemented as a priority queue that reorders contained URLs such that more promising pages are visited first, the crawler is said to be *preferential* or a *best-first* crawler. To estimate whether a page is promising, different strategies can be applied, for instance, calculation of PageRank (cf. Section 2.2.1 and [Page et al., 1999]).

For *fetching* the Web data, the "crawler acts as a Web client [i.e.] it sends an HTTP request to the server hosting the page and reads the response" [Liu, 2007], p.277. Typically, fetching of data is performed in parallel by multiple threads. "The frontier manager can improve the efficiency of the crawler by maintaining several parallel queues, where the URLs in each queue refer to a single server. In addition to spreading the load across many servers within any short time interval, this approach allows to keep connections with servers alive over many page requests, thus minimizing the overhead of TCP opening and closing handshakes" [Liu, 2007], p.286. Furthermore, issues like slowly or not responding servers, large files, broken links, or (cyclic) redirections have to be handled by the crawler.

For the tasks proposed, a *music-focused* crawler, i.e., a crawler biased towards music-specific Web pages, is the tool of choice. Alternatively, to obtain Web pages about a certain topic, one may use a *universal* crawler, i.e., a crawler that aims to index as many pages as possible, such as the crawlers used by Web search engines, and constrain pages to the domain of interest when accessing the harvested data (e.g., by including additional keywords within queries). In the present work, the latter approach is chosen, i.e., instead of crawling the Web and maintaining an index, the large-scale indexes of (commercial) Web search engines are used. More precisely, to obtain Web data that can be utilised in the proposed methods, the results returned by a Web search engine are downloaded and processed.

After fetching the data (and extracting the contained hyperlinks that are inserted into the frontier), several *preprocessing steps* are carried out before a document is stored and/or indexed. In a first step, all HTML tags are removed to obtain a plain text document. Next, all stopwords are removed, i.e., "frequently occurring and insignificant words in a language that help construct sentences but do not represent any content of the documents. Articles, prepositions and conjunctions and some pronouns are natural candidates." [Liu, 2007], p.199. Examples of English stopwords are *a*, *and*, *are*, *for*, *of*, *in*, *is*, *the*, *to*, or *with*. For indexing of Web pages as used in chapters 5 and 6, stopwords in six languages are removed: English, German, Spanish, French, Italian, and Portuguese.[3]

A typical next step to be performed is *stemming*. Stemming is the process of

[3]While stopword removal is useful in excluding unwanted terms, alternatively, a pre-defined list of desired expressions can be used. For indexing, only terms included in this list are considered. Such a vocabulary-based approach is used in Chapter 4.

removing prefixes and suffixes from a word, and thus reducing it to its stem. Since syntactic variations are removed, different forms of a word can be mapped to the same stem and facilitate retrieval of relevant documents. On the other side, important discriminating information may be lost. Throughout this thesis, no stemming is applied. Finally, all letters are converted to lower case, before the text is split up into a sequence of single word chunks where any white space, line break, or punctuation mark serves as word delimiter (a method commonly referred to as *bag-of-words*-modelling). The resulting sequence is then stored in a data structure that can be accessed efficiently at the word level, such as an *inverted index*.

3.1.2 Inverted Index

An inverted index I provides a lookup-table where for each distinctive term $t \in T$ that occurs in a collection of documents (in this case, Web pages), a list of all documents p that contain that term is provided. The reason for this method is that for retrieval (where queries consist of a sequence of keywords), documents that contain the query words can be found easily. Furthermore, in a *full inverted index*, additional information is stored with each document entry, such as number of occurrences of the corresponding term (useful for TF·IDF calculation, see Section 3.1.3) or positions of the term occurrences in the document (to allow searching for phrases, i.e., exactly matching sequences of words). To speed up calculation of IDF, for each term also the number of documents that contain the term (DF) can be stored explicitly. A toy example of an inverted index can be seen in Figure 3.1. In this example, three documents are indexed after stopword removal and conversion to lower case. For each of the indexed terms, a list of *postings* (which consist of a document ID, the frequency of occurrence of the term in that document, and the positions of these occurrences) is given (cf. [Liu, 2007], p.205). Note that (apart from the discarded stopwords), the documents can be reconstructed from the information stored in the index. However, usually, that is not of interest as the original documents are typically stored elsewhere and the index is only needed to efficiently find entries that satisfy the constraints (keywords) of a given query.

An additional benefit of inverted indexes is the compression of the data. First, multiple occurrences of terms within documents or across documents do not have to be stored redundantly due to the indexing. Furthermore, indexes can be compressed themselves, e.g., by storing only offset positions instead of absolute positions of term occurrences. When applying additional compression techniques for variable-length integers, offset storage can make a significant impact on the total index size, allowing even full indexes to reside in memory and therefore enhance performance (cf. [Liu, 2007], p.209*ff*). A comprehensive discussion of inverted indexes can be found in [Zobel and Moffat, 2006].

For retrieving relevant documents from an index, the query has to be processed in the same way as the documents. That is, conversion of all letters to lower case, splitting of words at punctuation marks, etc. has to be applied to the query string. To avoid unnecessary overhead, stopwords should also be removed. After this step, the index vocabulary has to be searched for the keywords contained in the query. In this step, efficiency is also important and can be achieved, e.g., by applying binary search on the lexicographically sorted vocabulary. For single keyword queries, the

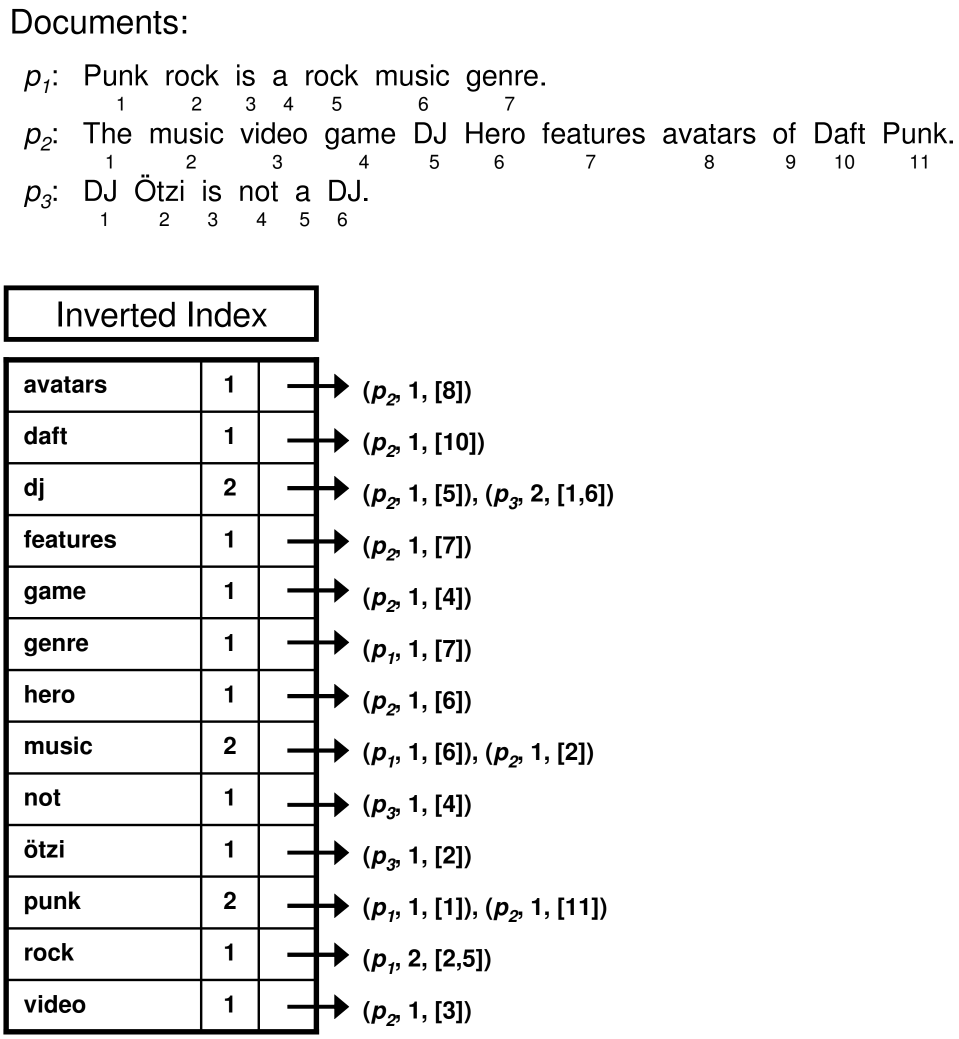


Figure 3.1: Example of an Inverted Index. Prior to indexing, stopword removal and lower case conversion are applied to the documents. For each term, the number of documents containing the term and a list of postings is stored. The format of a posting is (p_i, tf, [pos]), where p_i is the document identifier, tf the number of occurrences of the corresponding term in document p_i, and pos the list of positions at which the term occurs in p_i.

necessary step consists only in finding all documents that are associated with the keyword. For instance, querying the index from Figure 3.1 with the query *DJ* directly results in a response consisting of documents p_2 and p_3. (The ordering of the resulting documents is subject to the methods discussed in Section 3.1.3). For a composite query like *DJ Music* where the presence of multiple keywords is treated as an implicit conjunction, the intersection of the result sets of the single queries *DJ* (again, p_2 and p_3) and *Music* (p_1 and p_2) has to be calculated (p_2). When searching for phrases (often indicated by double quotes), in addition, the positional information of the indexed terms has to be considered. However, the condition that all retrieved documents must contain all query terms is often weakened to allow for retrieval of more documents (implicit disjunction of keywords). In this case, in the above mentioned example, the query *DJ Music* would result in retrieval of all three Web pages. Upon retrieval, documents are ranked according to relevance as described next.

3.1.3 Term Weighting and Document Ranking

When querying a system, it might not only be of interest to find all documents that fully satisfy all constraints, but also to obtain a ranking in which the most relevant documents with respect to the query are listed first.

In the above mentioned bag-of-words-model, a document is represented as the set of occurring terms (ordering of words in the corresponding text is not considered). For each term t occurring in a document p, a weight $w(t,p) > 0$ is assigned. If t does not occur in p then $w(t,p) = 0$. Each document p can therefore represented as a weight vector $\vec{p_j} = \langle w(t_1,p_j), w(t_2,p_j), ..., w(t_n,p_j)\rangle$ of the vocabulary vector $\vec{t} = \langle t_1, t_2, ..., t_n\rangle, t_i \in T, n = |T|$. Since every document is now represented by a vector, this model is also referred to as *vector space model*.

The computation of $w(t,p)$ depends on the retrieval model. The most common method is the calculation of a TF·IDF variant. The general scheme of all TF·IDF approaches is

$$w_{tfidf}(t,p) = tf(t,p) \cdot idf(t) = tf(t,p) \cdot log \frac{|P|}{df(t)}. \tag{3.1}$$

where P is the set of all documents indexed.

The idea behind this is that a high weight should be assigned if a term occurs frequently within a document (high TF). However, if a term occurs in many documents (high DF), it is not very discriminative and should therefore not have as much influence on the weighting (leading to a low IDF). Several versions of TF·IDF have been proposed. One of them is the "ltc" variant (cf. [Salton and Buckley, 1988] and [Debole and Sebastiani, 2003]):

$$w_{ltc}(t,p) = \begin{cases} (1 + \log_2 tf(t,p)) \log_2 \frac{|P|}{df(t)} & \text{if } tf(t,p) > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

Using the vector space model, it is possible to measure similarity between documents as well as similarity between a query q [4] and a document p, for instance, by calculating their *cosine similarity*:

$$sim_{cos}(q,p) = \frac{\vec{q} \cdot \vec{p}}{||\vec{q}|| \, ||\vec{p}||} = \frac{\sum_{t \in T} w(t,q) \cdot w(t,p)}{\sqrt{\sum_{t \in T} w(t,q)^2} \cdot \sqrt{\sum_{t \in T} w(t,p)^2}} \tag{3.3}$$

By measuring the angle between two document vectors, normalisation of vectors prior to comparison is not necessary. In case of another similarity measure (e.g., *Euclidean distance*), normalisation maybe required to remove the influence of different documents lengths on the weight scoring function.

[4] A query can simply be seen as a short document.

When retrieving documents, the results set can be simply ranked according to the similarities between query and documents. However, dealing with large document collections (such as the Web), it may be inefficient to calculate all these steps to obtain a relevance ranking for the documents. Therefore, simplified *scoring functions* may be calculated instead. For instance, in the Lucene indexer, the raw score of document p for query q is calculated as (cf. [Gospodnetić and Hatcher, 2005]):

$$score(q,p) = \sum_{t \in q} \left[\sqrt{tf(t,p)} \cdot idf_{luc}(t) \cdot boost \cdot lengthNorm \right] \tag{3.4}$$

$$idf_{luc}(t) = log \frac{|P|}{df(t)+1} + 1 \tag{3.5}$$

The factors *boost* and $lengthNorm$ are specific to each *field* in a Lucene index. In principle, an indexed document can have multiple fields, for instance, a Web page can be split into distinct fields such as title, URL, and text body. When querying, the query is compared to each of these fields separately and yields a score for each field. For assessing the overall relevance of a document, all field-specific scores of the document have to be combined, typically by summing up. In Equation 3.4, the value of *boost* can be adjusted manually for each field to explicitly raise importance of certain fields. The default setting (that is also used here) is $boost = 1$. The value of $lengthNorm$ is determined automatically and depends on the size of the term set in a field, i.e., here $lengthNorm = 1/\sqrt{|T|}$. Since only one field is considered in this work, i.e., the contents of the text body, both parameters are constant factors that do not influence the relevance ranking of documents.

Note that in the Lucene API definition as of version 2.4.0, for score calculation, additional factors, e.g., to normalise scores over different queries for comparability or to boost terms in the query, are incorporated. For details about these, the reader is referred to the API definition[5].

Another important source for relevance ranking is link structure (cf. Section 2.2.1). In the remainder of this thesis, for retrieval of data, this information is not incorporated, i.e., for index querying in chapters 5 and 6, the relevance ranking of documents is done only based on the Lucene scoring scheme.

3.1.4 Text Categorisation

This section deals with text categorisation, i.e., automatically assigning a text document to categories (*classes*) defined through example documents (*training set*). The task consists in extracting features from the given examples and to train a classifier that is then capable of generalising from the learned examples, i.e., being able to assign also new, previously unseen documents to the correct class (*supervised learning*). This can be used, for example, to automatically classify the content of Web pages into categories like *politics*, *sports*, *food*, or *music*, to name but a few. Such a component is essential, e.g., to build focused crawlers, where higher crawling priority has to be given to pages dealing with a specific topic (cf. Section 3.1.1).

The scenario in which this technique is required in the context of this thesis is the following: Not all of the Web documents retrieved using a Web search engine

[5] `http://lucene.apache.org/java/2_4_0/api/org/apache/lucene/search/Similarity.html`

are necessarily relevant and may therefore introduce errors to the subsequent steps. From a given set of such Web pages (instances of class “negative”) and also of Web pages that do not introduce errors (instances of class “positive”), a classifier can be trained to decide for new Web pages, whether they should be indexed (if classified as “positive”) or discarded (if classified as “negative”).

As features (*attributes* in machine learning terminology) for the Web pages, for instance, TF·IDF vectors as described in Section 3.1.3 can be utilised. However, “a major characteristic, or difficulty, of text categorization problems is the high dimensionality of the feature space. The native feature space consists of the unique terms (words or phrases) that occur in documents, which can be tens or hundreds of thousands of terms for even a moderate sized text collection. This is prohibitively high for many learning algorithms.” [Yang and Pedersen, 1997]. Therefore, typically, feature selection approaches such as the χ^2-test are applied to find the most discriminative features and to reduce the dimensionality of the feature space (cf. Section 6.2) “to make the use of conventional learning methods possible, to improve generalization accuracy, and to avoid ’overfitting’.” [Joachims, 1998]. A comparison of different strategies for feature selection in text categorisation can be found in [Yang and Pedersen, 1997].

Due to the nature of text categorisation tasks, i.e., high-dimensional input space, many relevant features, sparse document vectors ($w(t, p) = 0$ for most t and p), and, related to the sparsity, linear separability of categories, *Support Vector Machines* (SVMs, [Vapnik, 1995]) are a particularly good choice to serve as classifier (cf. [Joachims, 1998]). However, since SVMs are computationally complex, another type of classifier is used, namely the *Random Forest* classifier described next.

Random Forest Classifier

A Random Forest (see [Breiman, 2001]) is a classifier that consists of multiple *decision trees* and predicts the class that is obtained by voting on the predictions made by the individual trees (*ensemble classifier*). Before discussing this classifier in more detail, a short explanation of decision tree learning is given.

To build a decision tree for classification of (e.g., TF·IDF-based) instances, a divide-and-conquer strategy is applied. For each attribute a (i.e., each feature dimension), it is calculated how well a is suited to predict the target classes. In the scenario described above this translates to the question how well “positive” and “negative” training examples can be separated by just considering a. Typically, an entropy-based criteria is used for this calculation. Among all attributes, the most discriminative is chosen and the threshold (i.e., the decision boundary) that best separates positive from negative examples is determined. Using this, a decision node is added to the decision tree and the set of training examples is split into corresponding subsets. For each of the subsets, children nodes are added by repeating this step recursively on the remaining attributes until no more attributes remain or another termination criteria is fulfilled. To allow for better generalisation of the classifier, leaf nodes may be removed after training (*pruning*). For classification of instances, the decision tree is traversed top-down (i.e., starting from the root) according to the attribute values of the instance until a leaf is reached. The (most frequent) class in the leaf is then the predicted class of the instance. For a detailed description of a widely used decision tree classifier (*C4.5*) see [Quinlan, 1993].

The Random Forest classifier additionally incorporates the principles of *bagging* and *random feature selection* for learning the decision trees. Bagging refers to drawing multiple bootstrap samples, i.e., drawing samples with replacement, from the training examples, and training an ensemble of classifiers using the bootstrap samples. Random feature selection refers to finding the best predictor (i.e., the most discriminating attribute) from a randomly selected subset of attributes rather than from all attributes.

The principle algorithm for training and utilising a Random Forest is as follows (taken from [Liaw and Wiener, 2002]):

1. "Draw n bootstrap samples from the original data.

2. For each of the bootstrap samples, grow an unpruned classification [...] tree, with the following modification: at each node, rather than choosing the best split among all predictors, randomly sample m of the predictors and choose the best split from among those variables. [...]

3. Predict new data by aggregating the predictions of the n trees"

The main reasons Random Forest has been chosen over SVMs for classification in this thesis are the efficiency for learning from large training sets with a high feature dimensionality and the efficiency for classification of new instances. For implementation of the methods in Section 5.4.2.2, the Random Forest classifier from the WEKA package (see [Hall et al., 2009]) is applied with $n = 10$ trees. In addition, to compensate for unbalanced training sets (e.g., significantly more negative than positive examples) a *cost-sensitive meta classifier* is applied. If such a meta-classifier is used in conjunction with a Random Forest classifier, the generated trees are more sensitive to the more important class (in this case, the under-represented class). Usually, this is done by resampling the training data such that the training set is more balanced (with respect to the "costs" of the classes). The implementation from the WEKA package is also utilised for this purpose.

3.1.5 Text Alignment

Another approach to eliminate potential sources of errors is to identify and remove redundant parts in Web pages before indexing. Since Web pages often contain text sections not related to the actual content of the page, e.g., navigation bars, standard text indexing and ranking methods may be easily fooled by this data, leading to the inclusion of possibly irrelevant pages into responses to queries.

For finding redundant parts, a method originating from bioinformatics is applied: *multiple sequence alignment* (MSA). MSA is originally intended to arrange several DNA sequences (represented as sequences of single characters) to identify maximum overlaps and regions of similarity. For alignment, gaps have to be inserted into the sequences. The alignment process can be influenced by choosing rewards and penalties for correctly aligned characters, alignments with gaps, or misaligned characters.

The MSA method can be also applied to extract lyrics from multiple Web sources by matching coherent parts and preserving overlapping segments by [Knees et al., 2005]. In the present noise removal scenario, the overlapping segments are going to be deleted. The idea of performing multiple sequence alignment for identification of redundant Web data has also been presented, e.g., by [Zhai and Liu, 2005] or [Herscovici et al., 2007].

In the following, standard approaches for MSA are reviewed and adapted for alignment of text documents. First, a method to find optimum alignments between two sequences is introduced. This is followed by a heuristic that allows for alignment of more than two sequences. Finally, the identification of redundant parts is explained.

Needleman-Wunsch algorithm

Prior to the alignment, Web page contents have to be converted into a sequence of words. To align two word sequences, the Needleman-Wunsch algorithm can be used [Needleman and Wunsch, 1970]. This algorithm is based on dynamic programming and returns the globally optimal alignment of two strings for a given scoring scheme. For the task at hand, it is preferred to reward matching pairs of words with a high score (i.e., 10) and to penalise the insertion of gaps with a low value (i.e., -1). For mismatching words, a score of 0 is assigned. Using this configuration, the algorithm is expected to find large coherent segments and prefers to align single non-matching words instead of inserting gaps.

The algorithm itself uses a two-dimensional array MAT of size $(n+1)\times(m+1)$, where n is the length (the number of words) of the first sequence A and m the length of the second sequence B. The extra column and the extra row in the array are necessary to enable gaps at the beginning of each sequence. Every entry $MAT_{i,j}$ is interpreted as the optimal score of the partial alignment of $a_1, ..., a_i \in A$ and $b_1, ..., b_j \in B$. Thus, $MAT_{n,m}$ contains the score for the optimal alignment of A and B.

Entries of MAT are computed recursively. To calculate $MAT_{i,j}$, the optimal choice for the next alignment step is made by examining the three following cases (in the given order):

1. Alignment of a_i and b_j. This is equivalent to a diagonal step to the lower right in the array. Thus, the score at position $MAT_{i,j}$ is determined as the sum of $MAT_{i-1,j-1}$ and the score gained through alignment of a_i and b_j.

2. Alignment of a_i with a gap. This is equivalent to a step down. The score at position $MAT_{i,j}$ is then determined as the sum of $MAT_{i-1,j}$ and the penalty for gap insertion.

3. Alignment of b_j with a gap. This is equivalent to a step to the right. The score at position $MAT_{i,j}$ is then determined as the sum of $MAT_{i,j-1}$ and the penalty for gap insertion.

Considering these three options, the one to achieve the highest score is chosen. Substituting the values chosen, this results in

$$MAT_{i,j} = max \begin{cases} MAT_{i-1,j-1} & + \; s(a_i, b_j) \\ MAT_{i-1,j} & - \; 1 \\ MAT_{i,j-1} & - \; 1 \end{cases} \tag{3.6}$$

…	it's	showtime	-	for	dry	climes	and	bedlam	is	dreaming	of	rain	when	the	hills	…
…	its	show	time	for	dry	climes	and	bedlam	is	dreaming	of	rain	when	the	hills	…
…	it's	showtime	-	for	dry	climes	and	bedlam	is	dreaming	of	rain	when	the	hills	…
…	it's	showtime	-	for	drag	lines	and	bedlam	is	dreamin'	of	rain	when	the	hills	…
…	it's	showtime	-	for	dry	climes	and	bedlam	is	dreaming	of	rain	when	the	hills	…

Figure 3.2: Example of a multiple text alignment on song lyrics (*Los Angeles is burning* by *Bad Religion*), cf. [Knees et al., 2005]. The four rows on the top are word sequences extracted from the Web, the row at the bottom is the result obtained with any threshold value t below 0.75. While in this case the objective is to identify overlapping parts to extract the lyrics, for noise removal on Web pages, redundant parts should be identified and discarded.

where

$$s(x,y) = \begin{cases} 10, & \text{if x=y,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

Before performing the recursive procedure line-by-line, starting at i=1 and j=1, the array has to be initialised with

$$\begin{aligned} &MAT_{0,0} = 0, \\ &MAT_{i,0} = MAT_{i-1,0} - 1 \quad \text{for i=1,...,n, and} \\ &MAT_{0,j} = MAT_{0,j-1} - 1 \quad \text{for j=1,...,m.} \end{aligned}$$

After computation of all array entries, a trace back phase is necessary to determine the actual alignment from the scoring matrix. Starting from $MAT_{n,m}$ and depending on the origin of the score in the entries, the alignment is built backwards until $MAT_{0,0}$ is reached.

Multiple Sequence Alignment

The principle of the Needleman-Wunsch algorithm is theoretically easily extendible to more than two sequences. However, for two sequences this algorithm already uses $O(n \cdot m)$ in space and time. For every additional sequence the effort is further multiplied by the length of the sequence and is thus not practicable. To circumvent this problem, heuristics have been proposed to allow multiple sequence alignment with reasonable costs based on pairwise sequence alignment. Here, a hierarchical alignment, as proposed by [Corpet, 1988], is applied.

In the following, *row* is used to denote a sequence within a completed alignment, *column* denotes the list of words that are aligned together on a position in the alignment (cf. Figure 3.2), *length* is used to refer to the number of words in a row (i.e., the number of columns in an alignment), and *depth* refers to the number of sequences in a column. For example, the alignment in Figure 3.2 has depth four.

The hierarchical multiple sequence alignment algorithm works as described next. For all pairs of sequences, pairwise alignment using the Needleman-Wunsch algorithm is performed. The pair achieving the highest score is aligned. This step is performed again on the remaining sequences, until all are aligned (in case of an odd number of sequences, the last remains unaligned). The principle of pairwise alignment with respect to the highest score is then again applied to the group of aligned sequences, until only one alignment remains. For being able to perform pairwise alignment on pre-aligned sequences consisting of multiple rows, the Needleman-Wunsch algorithm has to be adapted. The basic idea is that words that already have been aligned, remain aligned. Thus, the algorithm must be capable of comparing columns of words, instead of single words. This is achieved with a simple modification in the scoring scheme:

$$MAT'_{i,j} = max \begin{cases} MAT'_{i-1,j-1} & + s_{col}(a'_i, b'_j) \\ MAT'_{i-1,j} & - 1 \\ MAT'_{i,j-1} & - 1 \end{cases} \tag{3.8}$$

where a'_i is the i^{th} column in alignment A' with depth k, b'_j the j^{th} column in alignment B' with depth l, and

$$s_{col}(x, y) = \sum_{i=1}^{k} \sum_{j=1}^{l} s(x_i, y_j), \tag{3.9}$$

with x_i denoting the i^{th} word in column x, and y_j the j^{th} word in column y. As can be seen, the score for aligning columns is simply given by the sum of the scores between all combinations of words from both columns. Note that the penalty score for gap insertions remained constant. Since this is now a score for insertions of columns consisting solely of gaps, in fact the penalty has been reduced because it is independent of the depth of the sequences. Thus, the idea of high rewards for matches and small penalties for gapping is further enforced for reasons explained above.

Identifying Redundancies

Given the multiple text alignment, redundant parts can be identified. To this end, every aligned column is examined for the most frequent word w. If the column's most frequent word is a gap, the column is skipped. Additionally, a threshold parameter t is introduced to determine if w is considered to be a redundant word. If the ratio of occurrences of w in the column and the depth of the alignment is below t, then the column is skipped too. For applying the method to noise removal, a value of $t = 0.6$ is chosen. As a result, several sequences of redundant words are obtained (columns to be skipped serve as delimiters to identify the boundaries of coherent segments). All sequences consisting of only one word are discarded. The remaining sequences are sorted according to length and, starting from the longest sequence, all occurrences of the sequences are deleted from all corresponding documents.

3.2 Audio Similarity

In this section, different methods to calculate content-based similarity between music tracks are reviewed. Section 3.2.1 deals with timbre-related similarity derived from MFCC features. Since MFCC-based methods suffer from some undesirable properties, Section 3.2.2 reviews a rank-based method to alleviate these drawbacks.

Section 3.2.3 deals with another type of audio features, namely Fluctuation Patterns that model rhythmic aspects. Finally, Section 3.2.4 reviews an approach to combine MFCC-based similarity and Fluctuation-Pattern-based similarity. A more detailed explanation of the methods discussed in this section can be found in [Pohle, 2009].

3.2.1 MFCC-based Spectral Features

MFCCs (Mel Frequency Cepstral Coefficients) are a standard signal processing technique, originally developed for automatic speech recognition [Rabiner and Juang, 1993], but also effective in music-related tasks (cf. [Logan, 2000]). MFCCs describe the spectrum of a frame and hence aspects related to timbre. Apart from other perceptually motivated transformations, the non-linear frequency resolution of the human auditory system is modelled by transforming the spectral representation of the frames to the Mel frequency scale. Another central property of MFCC-based approaches is that information on the temporal order of frames is discarded, i.e., all frames are treated equally, irrespective of their position within the piece ("bag-of-frames", cf. "bag-of-words" in Section 3.1.1). For music piece modelling, the resulting set of frames is summarised by clustering and/or calculating descriptive statistics (cf. [Pampalk, 2006]).

To calculate MFCCs, the signal "is analysed by converting it into mono format at 22050 Hz sample rate. [...] The signal is cut into frames of length 512 samples (with a hop size of 512 samples, $f_0 = 43.1$ Hz), and each frame is represented on the perceptually motivated mel scale by calculating the power FFT and subsequently applying a triangular-shaped mel filterbank with 36 bands. MFCCs are approximated by calculating the discrete cosine transform (DCT) on the logarithm of this representation". [Pohle, 2009], p.33. As proposed by [Aucouturier and Pachet, 2002b], 19 MFCCs are calculated on each frame.

In early work for music retrieval, the distribution of MFCC-transformed frames is described via a global supervised tree-based quantisation and the resulting features are used to retrieve short samples of sounds or tracks from the same artist [Foote, 1997]. A comparable, but more effective idea is presented in [Seyerlehner et al., 2008]. [Logan and Salomon, 2001] apply a k-means clustering to each song's distribution of MFCC-features. Every cluster is described by mean, covariance and a weight proportional to the number of frames belonging to that cluster. The set of k clusters is denoted the "signature" of the song. For distance computation of two songs, signatures are compared using the Earth Mover's Distance [Rubner et al., 1998].

A similar approach is proposed by [Aucouturier and Pachet, 2002a]. For clustering, a Gaussian Mixture Model (GMM) initialised with a k-means clustering is applied (cf. [Bishop, 1995]). A GMM models a given distribution as weighted sum of k Gaussian probability density functions and is trained using the EM-algorithm (Expectation-Maximization algorithm). To calculate the similarity of two songs, samples are drawn from each GMM and the likelihood of these samples given the other GMM is calculated. A symmetric similarity estimation is obtained by averaging the resulting likelihoods. According to [Mandel and Ellis, 2005] and [Pampalk, 2006], it is possible to accelerate this algorithm by a factor of about 20, while classification accuracies remain almost the same. Instead of using a GMM with multiple Gaussian components, one track is described by one Gaussian only (using mean and the full covariance matrix). The models of two songs can then be directly compared by calculating a symmetrised Kullback-Leibler divergence on the models. The distance between these models (and thus their corresponding music pieces) is denoted by $dist_G$.

As pointed out in [Aucouturier and Pachet, 2004], although being in general the most powerful signal-based approach to model similarity, all variants of Gaussian modelling of MFCCs are limited in that they exhibit what is called a "glass-ceiling" in terms of classification accuracy, irrespective of the chosen parameters (i.e., number of chosen MFCCs, k, distance computation). Furthermore, these approaches have a strong tendency to develop "hubs", i.e., some models/songs are significantly more often to be found among other's nearest neighbours than expected (also referred to the "always similar problem"). Since other types of similarity measures do not exhibit this "hub problem" – or at least have different hubs – while also representing different musical aspects, combined similarity measures can alleviate this problem (see Section 3.2.4). Another possibility of dealing with this issue is to perform post-processing on the obtained distance matrix, as explained in Section 3.2.2.

3.2.2 Post-Processing the Distance Matrix

As described in [Aucouturier, 2006] and [Pohle et al., 2006], the Kullback-Leibler divergence has some undesirable properties. For example, it can be observed that some particular pieces, so called hubs, are frequently "similar" (i.e., have a small distance) to many other pieces in the collection without sounding similar. On the other side, some pieces are never similar to others. Furthermore, the Kullback-Leibler divergence does not fulfil the triangle inequality.

To cope with these issues imposed by a distance measure that is no metric, [Pohle et al., 2006] propose a simple rank-based correction called *Proximity Verification* that replaces the distances in the distance matrix D with a rank-based measure. The entries of each row of the distance matrix D are sorted in ascending order, and each original entry of the row is replaced with its rank. The resulting distance matrix (denoted D_1 here) is transformed into the final matrix by adding the transpose (resulting in a symmetric matrix): $D_{PV} := D_1 + D_1'$. The resulting matrix has a better distribution of distances than the original distance matrix, reduces the "hubness", and seems to be better suited as input for subsequent steps such as clustering (cf. Section 3.3.1).

As a consequence of this modification, all subsequent steps can only utilise the ranking information of audio similarity, i.e., audio similarity information can only be used to the extent of whether a piece is most similar, second most similar, third most similar, etc. to a piece under consideration and not to which numerical extent the two pieces are considered to be similar.

3.2.3 Fluctuation Patterns

The rhythm-based Fluctuation Patterns (FPs) model periodicities in the audio signal. In the following, only the main steps in the computation of these features are sketched. For more details, the reader is referred to the original sources, i.e., [Pampalk, 2001] and [Pampalk et al., 2002].

The feature extraction process is carried out on short segments of the signal, i.e., every third 6 second sequence. In a first step, an FFT is applied to these audio segments. From the frequencies of the resulting spectrum, 20 critical-bands are calculated according to the bark scale. Furthermore, spectral masking effects are taken into account. In a next step, several loudness transformations are applied. As a consequence, the processed piece of music is represented by a number of feature matrices that contain information about the perceived loudness at a specific point in time in a specific critical band. In the following stage another FFT is applied, which gives information about the amplitude modulation. These so-called fluctuations describe rhythmic properties by revealing how often a specific frequency reoccurs. Additionally, a psychoacoustic model of fluctuation strength is applied since the perception of the fluctuations depends on their periodicity, e.g., reoccurring beats at 4 Hz are discerned most intensely. In a final step, the median of all Fluctuation Pattern representations for the processed piece is calculated to obtain a unique, typically 1,200-dimensional feature vector for each piece of music. These vector features can be used for similarity calculation, for instance, by calculating Euclidean distances (see Equation 4.4) or the cosine similarity sim_{cos} as defined in Equation 3.3. Following the naming convention by [Pohle and Schnitzer, 2007], for further steps (see Section 3.2.4), $dist_{FP}$ denotes the FP-based distance between two music pieces m and n defined as $dist_{FP}(m,n) = 1 - sim_{cos}(FP(m), FP(n))$, where $FP(m)$ represents the fluctuation pattern of a track m. Alternatively, FP vectors can be used as input for clustering algorithms, e.g., as a first step for generating music maps (cf. Section 3.3).

3.2.4 Combined Similarity

To obtain an improved audio similarity measure that incorporates multiple facets of music, [Pohle and Schnitzer, 2007] propose a combination of existing similarity measures. Their algorithm competed successfully in the "Audio Music Similarity and Retrieval" task of MIREX 2007[6] and therefore represents one of the world-leading signal-based similarity measures. Based on an initial approach for combination of MFCC- and FP-based similarity measures proposed by [Pampalk, 2006], [Pohle and Schnitzer, 2007] developed the following combination scheme:

To complement the single-Gaussian-MFCC-model-based distance $dist_G$ and the FP-based distance $dist_{FP}$, two additional FP-related features are computed: *Bass (FPB)* and *Gravity (FPG)* (for details see [Pampalk, 2006]). These two features are scalar and the distances between two songs, denoted by $dist_{FPB}$ and $dist_{FPG}$, respectively, can thus simply be calculated by subtracting the corresponding feature values. Note that, in contrast to Section 3.2.1, here, for the MFCC features, 25 MFCCs are computed on each frame. To obtain an overall distance value $dist_{comb}$ measuring the dissimilarity of two songs, all described distance measures are combined by a simple arithmetic weighting:

$$dist_{comb} = 0.7 \cdot z_G + 0.1 \cdot (z_{FP} + z_{FPB} + z_{FPG}) \tag{3.10}$$

where z_x is the value of $dist_x$ after *z-normalising*, i.e., standardising of values

[6] http://www.music-ir.org/mirex/wiki/2007:MIREX2007_Results

by subtracting the mean value of all distances originating from the same feature and dividing by the standard deviation. Finally, distances between two songs are symmetrised.

3.3 Music Maps

Music maps are an approach to automatical structuring of music collections based on features extracted from audio content. In the field of Music Information Retrieval, a frequently used technique is to apply a Self-Organizing Map (SOM) to arrange a music collection on a two-dimensional map. The SOM is a widely used technique for exploratory data analysis and visualisation of high-dimensional data sets. It is explained in Section 3.3.1.

The first approach to structure music collections using a SOM was presented by [Rauber and Frühwirth, 2001]. Since then, several extensions and alternatives have been presented as can be seen in Section 2.4.2. One of the most interesting among these is the *Islands of Music* visualisation technique by [Pampalk, 2001] that can also serve as an user interface to music repositories. The Islands of Music are reviewed in Section 3.3.2.

3.3.1 Self-Organizing Map

Being confronted with complex high-dimensional data without any class information, it is desirable to have methods to explore the data and to automatically find structures inherent to the data (*unsupervised learning*). An approach to uncover such intrinsic structures is clustering. “Clustering is the process of organizing data instances into groups whose members are similar in some way. A cluster is therefore a collection of data instances which are ’similar’ to each other and are ’dissimilar’ to data instances in other clusters.” [Liu, 2007], p.117–118.

The SOM is both a clustering method and a data projection method that organises and visualises multivariate data on a usually two-dimensional map (see [Kohonen, 2001]). This is achieved by performing clustering within the feature space and projecting data instances belonging to a cluster to a corresponding cell on a grid, i.e., the map. The grid structure of the map is also reflected by the clustering in feature space (see below). Therefore, data items that are located closely in the feature space should also be located closely on the map.

More formally, the SOM consists of an ordered set of map units, each of which is assigned a *model vector* in the original data space that represents the centre of a cluster. The set of all model vectors of a SOM is called its *codebook*. There exist different strategies to initialise the codebook; here, linear initialisation is applied, cf. [Kohonen, 2001]. For training, the batch SOM algorithm by [Tai, 1995] is adopted:

- In a first step, for each data item x, the Euclidean distance (Equation 4.4) between x and each model vector is calculated. The map unit possessing the model vector that is closest to a data item x is referred to as the *best matching unit* and is used to represent x on the map.

- In the second step, the codebook is updated by calculating weighted centroids of all data elements associated with the corresponding model vectors. This reduces the distances between the data items and the model vectors of the best matching units and also their surrounding units, which participate to a certain extent in the adaptations. The adaptation strength decreases gradually and depends on both distance of the units and iteration cycle. This supports the formation of large clusters in the beginning and a fine-tuning toward the end of the training. Usually, the iterative training is continued until a convergence criterion is fulfilled.

As a result, data items which are similar in the high-dimensional space are assigned to similar locations on the map. The resulting map allows for insights into the inherent structure and an overview of the distribution of the data.

The data instances input to a SOM are typically feature vectors since the SOM makes use of Euclidean distances for clustering. For instance, [Pampalk et al., 2002] uses Fluctuation Pattern vectors as input. Alternatively, similarity matrices, such as those originating from the MFCC-based similarity measures after post-processing (Section 3.2.2) or the combined similarity measures (Section 3.2.4), can be used if the j^{th} column of the similarity matrix is interpreted as vector representation of the j^{th} song.[7]

3.3.2 Islands of Music

The Islands of Music are a technique developed by [Pampalk, 2001] to visualise SOM-based music maps created from Fluctuation Patterns. The idea is to create appealing interfaces that build upon a metaphor of geographical maps to illustrate the distribution of music pieces on the music map. Thus, on an Islands of Music enhanced map, areas onto which only few pieces of music are mapped are indicated by blue regions (oceans), whereas clusters containing a larger quantity of pieces are coloured in green, brown, and white (hills, mountains, and snow, respectively).

To obtain a texture that reflects the (discrete) distribution of pieces on the SOM grid while appearing to represent a landscape, a so-called *Smoothed Data Histogram* (SDH) is calculated. An SDH creates a smooth height profile (where height corresponds to the number of items in each region) by estimating the density of the data items over the map. To this end, each data item votes for a fixed number of best matching map units. The selected units are weighted according to the degree of the matching. The votes are accumulated in a matrix describing the distribution over the complete map. After each piece of music has voted, the resulting matrix is interpolated in order to obtain a smooth visualisation. Finally, a colour model inspired by geographical maps is applied, to give the impression of an island-like terrain and to emphasise the resulting height profile.

[7] A superior method for creating music maps from Gaussian-modelled MFCC features has been recently described by [Schnitzer et al., 2010]. Instead of creating a similarity matrix and interpreting the columns as features, SOM training is performed directly on Gaussian music similarity features. This approach creates music maps that are less costly to compute and better preserve the original similarity topology. However, for proof-of-concept of the methods presented in Chapter 4, the (somehow flawed, but nevertheless) well established method that interprets the columns of a similarity matrix as feature vectors is applied.

Figure 3.3 shows an Islands of Music visualisation created with the CoMIRVA framework[8] developed at the Department of Computational Perception (for details see [Schedl et al., 2007]).

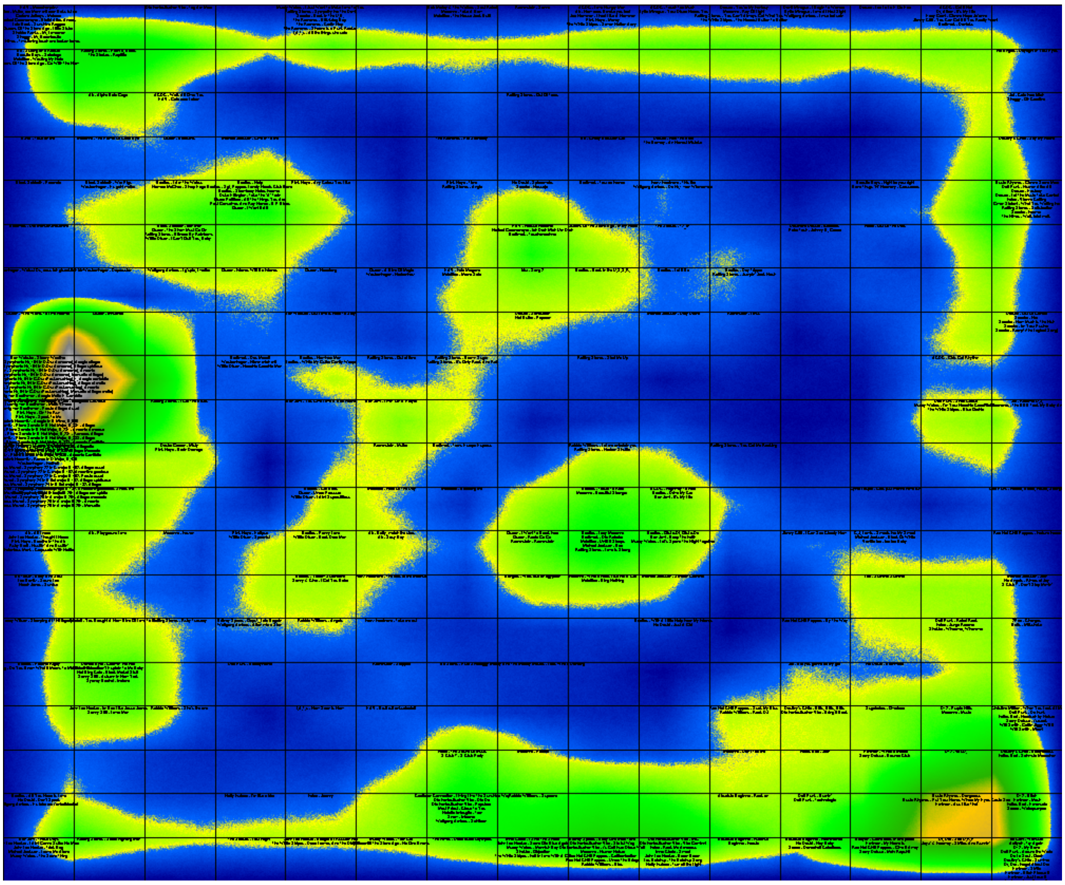


Figure 3.3: An Islands of Music visualisation created with the CoMIRVA framework.

[8] `http://www.cp.jku.at/comirva/`

Chapter 4

Automatically Deriving Music Labels for Browsing

A user interested in finding music – whether known or unknown – is not necessarily willing to formulate a search-engine-specific query. Instead, the available music collection might just be explored in an undirected manner. This corresponds to people's behaviour when searching for new music in (physical) record stores or when examining private music collections (cf. [Cunningham et al., 2004]). This process of undirected exploration is called *browsing*. As stated by [Baeza-Yates and Ribeiro-Neto, 1999], p.65, "in general, the goal of a searching task is clearer in the mind of the user than the goal of a browsing task." Hence, it is important to support the user in finding items of interest by providing guidance.

In this chapter, an approach to extend map-based browsing interfaces by means of semantic descriptors is presented. More precisely, audio-based music maps (cf. Section 3.3) are labelled with automatically extracted terms that describe the musical characteristics in the different regions of the map. The resulting *Music Description Map* can be very useful for browsing large music repositories, since the terms serve as landmarks on the map, allowing for better orientation. Furthermore, this technique is incorporated into *nepTune*, an interactive three-dimensional immersive interface, to support the user in exploring music collections. A schematic overview of the techniques elaborated in this chapter can be found in Figure 4.1.

The remainder of this chapter is organised as follows. In the next section, shortcomings of standard music maps are discussed and the development of techniques for meaningful labelling is motivated. Section 4.2 presents the technical details of the Music Description Map. Section 4.3 elaborates on the nepTune interface and the integration of the developed labelling techniques. Section 4.4 reports on evaluating the presented techniques. The chapter finishes with a discussion of the results (Section 4.5).

4.1 Motivation

Since digital music collections nowadays comprise vast amounts of pieces, automatic structuring and organisation of large music repositories is a central challenge. One approach to address this challenge is to perform clustering on features extracted from the audio-signal to create music maps (cf. Section 3.3). This method of structuring a collection in a two-dimensional layout allows then for browsing by examining different regions on the map. This can be accomplished by selecting the contained

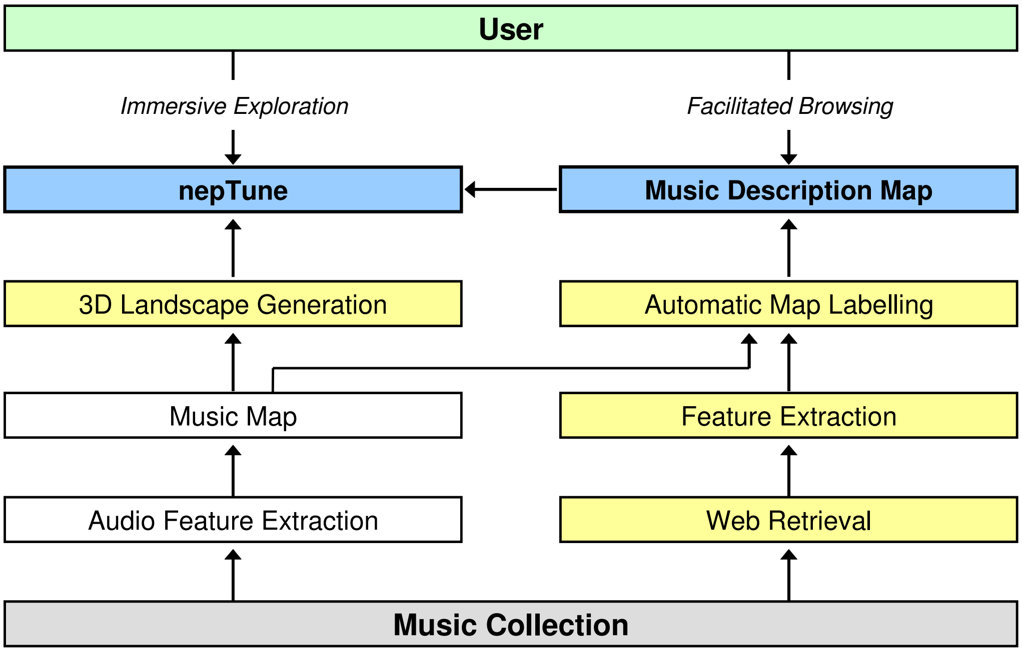


Figure 4.1: Schematic overview of the methods presented in this chapter. Yellow bars indicate the methods presented. The focus is put on the steps outlined on the right. Blue bars represent the final outcome of the presented research, i.e., the Music Description Map technique and the nepTune user interface.

pieces and listening to them. A drawback of this is that maps tend to get cluttered very easily when the pieces are represented by their meta-data, for instance by a combination of artist name and title of the track. An example of such a cluttered music map can be seen in Figure 4.2. To alleviate this, a possible alternative may consist in introducing a level of abstraction by not displaying individual track descriptors on each map unit but only the names of the most frequently occurring artists or the most representative genre descriptors (cf. Figure 4.4). For the latter, genre information has to be available, however. Furthermore, genre descriptors are a particularly problematic choice since well-distinguishable and thus easily accessible genre taxonomies mostly consist of a very limited vocabulary while representing only broad categories (cf. [Pachet and Cazaly, 2000]). To allow for a more diverse set of descriptors and a possibly more specific association (while not requiring the availability of explicit stylistic annotations of collections) the approach to be presented here makes use of descriptors automatically derived from the Web to find adequate labels for specific regions on music maps and improve orientation. Hence, this approach exploits context-based data to augment an interface structured by means of content-based similarity.[1]

[1]While most work that deals with incorporating both music content and context use contextual information to learn the "meaning" of certain acoustic properties (i.e., pursue a bottom-up approach to overcome the so-called "semantic gap", cf. sections 2.1.3 and 2.3), in this chapter, a top-down approach to find a mapping between signal and "meaning" is pursued. Instead of assigning descriptions to certain low-level characteristics of an audio signal, the here proposed technique

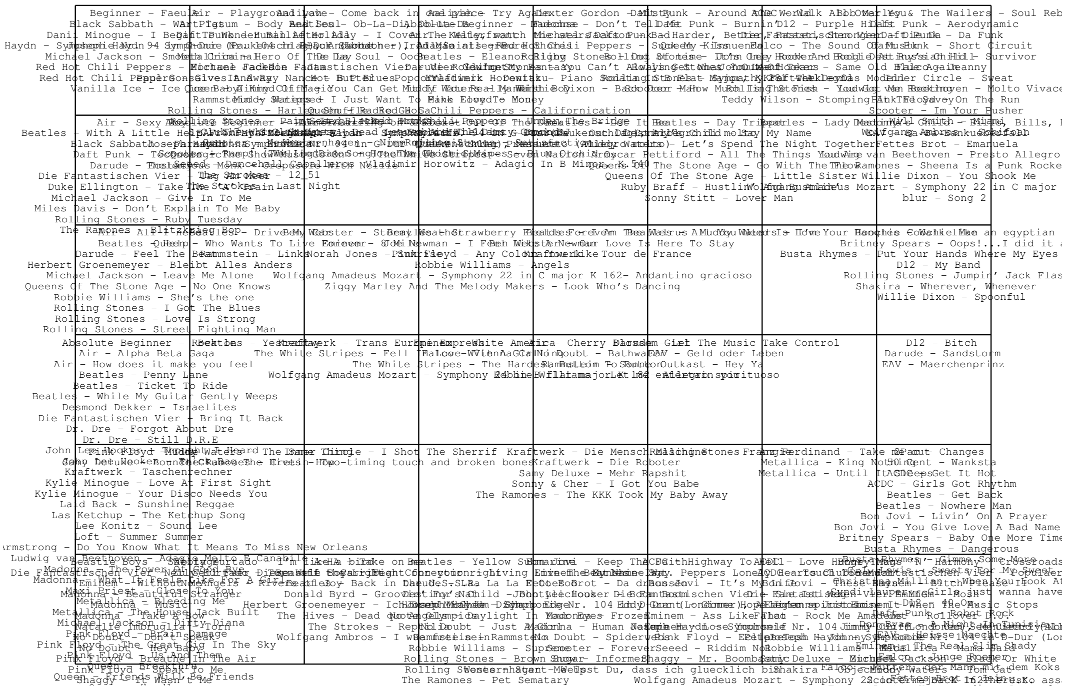


Figure 4.2: A very cluttered music map.

An additional contribution of the presented approach – although it is only but a first step in this direction – is Web-based description and indexing of music at the track level. While Web-based approaches are capable of capturing information about artists to a certain extent and thus usually perform well on the artist level (cf. Section 2.1.2.3), finer distinctions between different tracks of artists have not found their way into feature representations. Hence, another aim of this chapter is to specifically bridge the gap between signal-based organisation of music archives at the level of individual tracks and word descriptors of the work of individual musical artists. By obtaining a mapping between musical pieces in the audio feature space and the cultural context, this mapping could also be utilised for indexing of tracks and to allow for querying a music retrieval system by describing musical contents with familiar terms (cf. Chapter 5).

The presented technique can further be applied for augmenting nepTune, an advanced three-dimensional music browsing interface that creates a virtual landscape for exploration. Using the Music Description Map (MDM) technique, the music landscape can be enriched with meaningful information and also visual add-ons. Instead of displaying just meta-data on the landscape for orientation, the user can then choose to see words that describe the heard music or images that are related to this content. Thus, besides a purely audio-based structuring, nepTune offers more contextual information that may trigger new associations in the listener/viewer, thus making the experience of exploration more interesting and rewarding.

aims at describing groups of musical pieces, i.e., regions on a map, with culturally related terms.

4.2 Music Description Map

Although neither music maps nor the Music Description Map are bound to a specific audio similarity measure, first the process of creating the music maps that underlie the presented technique is described briefly. For audio feature extraction and similarity estimation, an MFCC-based method (cf. Section 3.2.1) has been applied, namely the similarity measure proposed by [Mandel and Ellis, 2005]. To deal with the inherent hub-problem discussed in Section 3.2.1, Proximity Verification as described in Section 3.2.2 is incorporated. Hence, a distance matrix with a better distribution of distances is obtained that seems to be better suited as input for the next step, namely training of a map using the SOM algorithm (see Section 3.3.1). Since the distance matrix contains the pairwise distances between tracks, whereas the SOM algorithm expects points in Euclidean space as input, each column of the similarity matrix is interpreted as one vector in Euclidean space, where the i^{th} row corresponds to the i^{th} song. As result of the SOM training step and the final assignment of each data point to its best matching unit, a music map, i.e., a two-dimensional grid that maps every piece in the collection to one of the contained cells, is obtained.

For the creation of a Music Description Map, three steps have to be performed that are elaborated in the following sections:

1. Retrieval of information from the Web to create term profile descriptions of the musical artists contained in the collection

2. Association of each track in the collection with the term characterisation of its corresponding artist and labelling of the SOM based on these representations

3. Finding similarly labelled units to detect and merge larger coherent regions

4.2.1 Artist Term Profile Retrieval

As mentioned before, while it is non-trivial to find specific information on certain songs, extracting information describing the general style of an artist is feasible (cf. Section 2.1.2.3). Usually, the acquisition of artist descriptors is realised by invoking Google with a query like *"artist" music review* and analysing the first 50 or 100 returned pages, by counting term frequency and document frequency for either single words, bi-grams, or tri-grams and combining them into the TF·IDF measure (cf. Section 3.1.3). For the purpose of creating an MDM, this approach is also applicable. However, downloading 100 pages for each artist consumes bandwidth and time and is not necessarily required. To speed up the artist profile extraction (which is crucial to allow for integration of the technique in time-critical applications like nepTune in the context of a public installation, cf. Section 4.3.1), the search for musical style is simplified by formulating the query *"artist" music style* and retrieving only Google's result page containing the top 100 hits (cf. [Knees et al., 2008b]). Besides links to the first 100 pages, the result page contains extracts of the relevant sections ("snippets") of the pages. Hence, instead of each individual Web page, only the result page — foremost the snippets presented — is analysed, reducing the effort to downloading and analysing only one Web page per artist. Another advantage of analysing only the "digest" of the artist-related pages is to incorporate only information from the most important sections of the Web pages, more precisely, the most relevant sections with respect to the query. Otherwise, structural analysis of each Web page would be necessary to avoid inclusion of unrelated text portions (e.g., from navigation bars, cf. Section 5.4.1.1). Further, to eliminate musically unrelated words, a dictionary approach with a reduced version of the vocabulary created by [Pampalk et al., 2005] is used. Thus, only occurrences of words or phrases that are contained in this domain-specific vocabulary T of 944 music-related terms are counted. The list of contained terms can be found in Table A.1. After obtaining, for each artist a, a term frequency representation $\vec{tf}_a = \langle tf(t_1, a), tf(t_2, a), ..., tf(t_n, a)\rangle$ of the vocabulary vector $\vec{t} = \langle t_1, t_2, ..., t_n\rangle, t_i \in T$ for each artist a, the important terms for each cluster are determined as described next.

4.2.2 Map Labelling

Having obtained artist-related term descriptors, a strategy to determine those words that discriminate between the music in one region of the map and music in another is needed. For instance the term *Music* is not a discriminating word, since it occurs very frequently for all artists; *Piano* would be a valuable word to indicate piano music, assuming piano music forms a distinct cluster on the map.

Considering the requirements imposed by the task, the SOM-labelling strategy proposed by [Lagus and Kaski, 1999] — originally developed to support the WEBSOM technique (cf. [Kaski et al., 1998]) — seems to be the best choice. In the context of clustering text collections, several other strategies for finding discriminatory labels have been proposed, among them the LabelSOM approach by [Rauber, 1999]. However, while Lagus' and Kaski's method determines the importance of descriptors for each cluster based on the contained items, the LabelSOM approach selects terms that represent the most relevant dimensions for assigning data to a cluster in the training phase of the SOM. As a consequence, Lagus' and Kaski's approach can also be used in "situations where the data of interest is numeric, but where some texts can be meaningfully associated with the data items", as the authors state in the conclusions of [Lagus and Kaski, 1999]. Hence, this strategy can be applied for the task at hand, i.e., to label a SOM trained on audio features with semantic descriptors extracted automatically from the Web.

In the heuristically motivated weighting scheme by [Lagus and Kaski, 1999], knowledge of the structure of the SOM is exploited to enforce the emergence of areas with coherent descriptions. To this end, terms from directly neighbouring units are accumulated and terms from a more distant "neutral zone" are ignored. The goodness G of a term $t \in T$ as a descriptor for unit u is calculated as

$$G(t, u) = \frac{\left[\sum_{k \in A_0^u} F(t, k)\right]^2}{\sum_{i \notin A_1^u} F(t, i)}, \tag{4.1}$$

where $k \in A_0^u$ if the L_1 distance of units u and k on the map is below a threshold r_0, and $i \in A_1^u$ if the distance of u and i is greater than r_0 and smaller than r_1. In the experiments presented in the following, values of $r_0 = 1$ and $r_1 = 2$ have been chosen. $F(t, u)$ denotes the relative frequency of term t on unit u and is calculated as

$$F(t,u) = \frac{\sum_{a} f(a,u) \cdot tf(t,a)}{\sum_{v} \sum_{a} f(a,u) \cdot tf(v,a)}, \tag{4.2}$$

where $f(a, u)$ gives the number of tracks of artist a on unit u. As a result, for each unit u, a term-goodness vector $\vec{g}_u = \langle G(t_1, u), G(t_2, u), ..., G(t_n, u)\rangle$ is obtained. [2]

For display of terms, all entries with $G < 0.01$ are ignored and at most 30 terms are selected to appear on a map unit (provided that there is enough space to display them). Furthermore, the font size of a term is set according to its score (cf. Figure 2.2). However, this approach can again lead to very cluttered maps. Another shortcoming is that many neighbouring units may contain very similar descriptions. Thus, one could easily happen "not to see the wood for the trees" when orienting on the map. Since the aim is to provide clearly arranged maps to make it *simpler* to find music, the next goal is to find coherent parts of the MDM and join them to single clusters.

4.2.3 Connecting Similar Clusters

To identify adjacent units with similar descriptors the following heuristic is applied: First, all units on the map are sorted according to $max_{\, t \in T} \{G(t, u)\}$, i.e., the maximum G value of the contained terms. Starting with the highest ranked unit, a recursive cluster expansion step is performed for all units. In this step, the adjacent four neighbours (i.e., those units with an L_1 distance of 1 to the unit under consideration) are examined for similar labels. The idea is to create a single vector representation that adequately reflects the vectors of both connected units. This is achieved by calculating *cosine normalised* versions of both units' description vectors and comparing them to a cosine normalised version of the vector obtained by adding both units' vectors. For comparison, *Euclidean distances* are calculated.[3] More precisely, when investigating whether to join clusters u and v, both units' term-goodness vectors, i.e., $\vec{g}_u$ and $\vec{g}_v$, are normalised to obtain the vectors $\hat{g}_u$ and $\hat{g}_v$, respectively, where $\hat{g}_u = \left\langle \hat{G}(t_1, u), \hat{G}(t_2, u), ..., \hat{G}(t_n, u) \right\rangle$ and $\hat{G}(t, u)$ is the cosine normalised goodness value calculated as

$$\hat{G}(t,u) = \frac{G(t,u)}{\sqrt{\sum_{s \in T} G(s,u)^2}}. \tag{4.3}$$

Both $\hat{g}_u$ and $\hat{g}_v$ are then compared with the normalised version $\hat{g}_{u+v}$ of the vector sum $\vec{g}_{u+v} = \vec{g}_u + \vec{g}_v$ by calculating the Euclidean distances $euc(\hat{g}_u, \hat{g}_{u+v})$ and

[2] To find the most discriminating terms for each cluster, experiments with the χ^2-test (cf. Equation 6.2) have also been made. The χ^2-test is a well-applicable method to reduce the feature space in text categorisation problems. In a final assessment, Lagus' and Kaski's approach has been chosen over the χ^2-test because it incorporates information on the structure of the map as well as yielding more comprehensible results.

[3] Alternatively, the cosine similarity could be calculated between the non-normalised vectors.

$euc(\hat{g}_v, \hat{g}_{u+v})$, with $euc(\vec{x}, \vec{y})$ defined as

$$euc(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4.4}$$

If both $euc(\hat{g}_u, \hat{g}_{u+v})$ and $euc(\hat{g}_v, \hat{g}_{u+v})$ are below a threshold (in the following, an empirically determined value of 0.4 is used), i.e., if the resulting vector $\vec{g}_{u+v}$ is sufficiently similar to the original ones and thus suitable to represent both original vectors, the candidate unit is admitted to the cluster. Furthermore, all units in the cluster are assigned the newly obtained term description vector (i.e., $\vec{g}_u \leftarrow \vec{g}_{u+v}$ and $\vec{g}_v \leftarrow \vec{g}_{u+v}$). Thus, as regions grow larger, corresponding descriptors will become more important globally. For all absorbed units, this procedure is repeated recursively until no more clusters can be joined.

4.2.4 Example

An example of an MDM with connected map units can be found in Figure 4.3. For comparison, the underlying SOM is depicted in Figure 4.4 with genre frequencies for individual units.

Taking a look at the MDM, one can easily identify the contained music at first sight or at least get a good impression of the collection (2,572 tracks from 7 genres). A more detailed examination reveals that Jazz has two major clusters – one consisting of *Piano* Jazz (together with Classical Music), and one consisting of *Trumpet* Jazz. Genres like Metal are also represented through more distinct information (e.g., *Gothic* vs. *Power Metal*). Adjectives (*energetic*, *percussive*, *aggressive*, etc.) can give information also to users unfamiliar with the presented style descriptors.

4.2.5 Further Enhancements

As can be seen in Section 4.2.4, pieces from culturally related music styles may form separated clusters on the map due to acoustic properties that enforce discrimination by the applied music similarity measure. Hence, similar Web descriptors may appear in different areas on the map. However, the Web descriptors assigned to each cluster by means of the MDM can also be considered as an alternative feature representation in a "semantic" term space. To further unveil these "culturogenic" similarities between clusters that are placed apart, the additional dimension of similarity may be visualised by an automatic map colouring approach.

One possibility to map similar term vectors to similar colours consists in applying a multidimensional scaling (MDS) approach to project each item into a colour space while preserving gradual differences in similarity as well as clear dissimilarities (cf. [Kaski et al., 1999]). Hence, similar colours on the map indicate similar music. Furthermore, for an interactive browsing interface, e.g., realised using a touchscreen, only the most discriminative words could be displayed statically to allow for a coarse orientation, while more distinct words appear as soon as corresponding regions are explored. Figure 4.5 depicts a screenshot of a prototype implementation of such a Colour MDM.

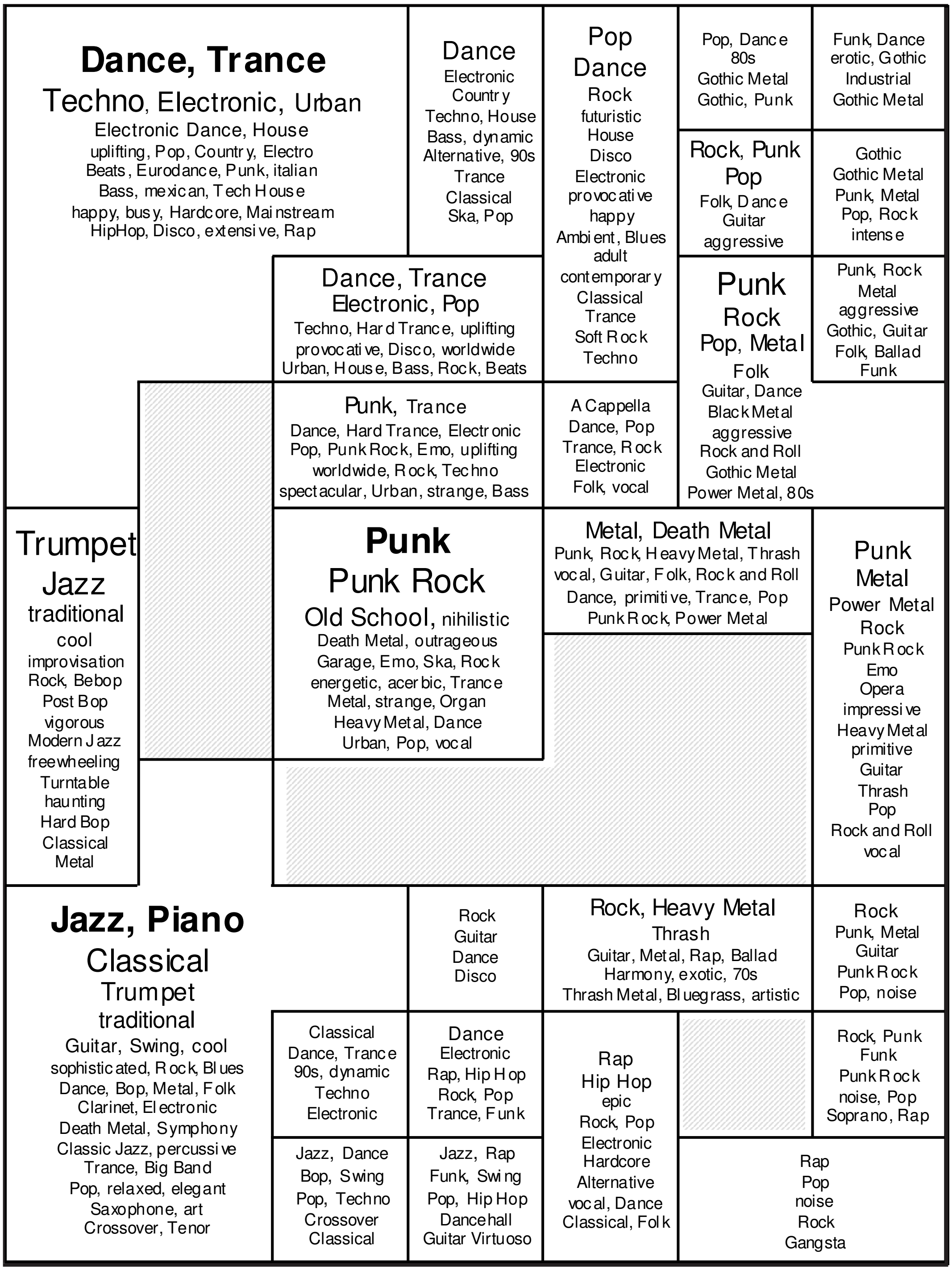


Figure 4.3: A Music Description Map with joined similar clusters. The displayed terms describe the musical style in the different regions of the underlying music map (7×10 SOM) depicted in Figure 4.4. The size of the terms reflect the importance of the descriptor for the corresponding region. For reasons of readability, line breaks have been edited manually.

Dance (7) Hip-Hop (3) Jazz Metal Punk	Dance (13) Hip-Hop (2) Jazz Pop	Dance (23) Hip-Hop (2) Jazz	Dance (15) Hip-Hop (4) Pop (4) Jazz (3)	Dance (144) Pop (113) Hip-Hop (50) Jazz (19) Punk (15) Metal (10)	Dance (14) Pop (6) Punk (3) Hip-Hop (2) Jazz (2) Metal (2)	Dance (61) Pop (23) Metal (13) Punk (8) Hip-Hop
Dance (4) Hip-Hop Pop	Dance	Dance (6) Hip-Hop Pop	Dance (2)	Dance (3) Metal Pop	Pop (3) Metal Punk	Dance (28) Pop (17) Metal (11) Punk (7) Hip-Hop (2) Jazz
Dance (7) Metal	Dance (2) Punk	Dance (5) Hip-Hop (2) Pop	Dance (3) Metal Pop Punk	Dance (6) Hip-Hop Pop	Metal Punk	Metal (23) Pop (21) Punk (14) Dance (12) Hip-Hop (2)
Dance (6) Hip-Hop	Dance (3)	Hip-Hop (2) Dance Metal Punk	Dance (2) Punk	Dance (4)	Punk (3) Metal	Metal (42) Punk (18) Pop (12) Dance (10) Hip-Hop (2)
Jazz		Punk	Metal (94) Punk (37) Dance (3) Pop (3)	Metal (2) Dance	Metal (3) Punk (3)	Metal (192) Punk (173) Pop (6) Dance (4) Hip-Hop
Jazz		Punk	Punk			Metal (2) Pop (2) Punk (2)
Metal (3) Jazz	Jazz (2)					Punk (4) Metal (2) Pop (2)
Classical (217) Jazz (129) Metal (3)	Jazz (3) Metal	Jazz (5) Hip-Hop	Dance (2) Pop (2)	Metal	Metal (2)	Punk (5) Pop (4) Metal
Jazz (23) Classical (4) Pop (2) Dance	Jazz (2) Dance	Dance (2)	Hip-Hop (3) Jazz (2) Pop (2) Dance	Hip-Hop (2) Pop		Pop (4) Punk (2) Hip-Hop Jazz
Jazz (123) Dance (10) Pop (6) Classical (5) Hip-Hop (3) Metal (3)	Jazz (30) Dance (7) Pop (4) Hip-Hop (3)	Jazz (24) Pop (14) Dance (10) Hip-Hop (6) Metal	Pop (59) Jazz (55) Hip-Hop (37) Dance (20) Metal (2) Punk	Hip-Hop (51) Pop (44) Dance (30) Jazz (21) Metal	Hip-Hop (34) Pop (26) Dance (25) Jazz (5) Metal (2)	Hip-Hop (23) Dance (12) Pop (12) Jazz (4) Metal (3) Punk

Figure 4.4: A 7×10 SOM trained on a collection containing 2,572 tracks (by 331 artists) assigned to 7 different genres.Tracks from Jazz, as well as from Classical, tend to cluster at the (lower) left, Dance at the top. Centre and right side are dominated by Punk and Metal. Hip-Hop is mainly found at the bottom. Pop occurs frequently in conjunction with Dance and Hip-Hop.

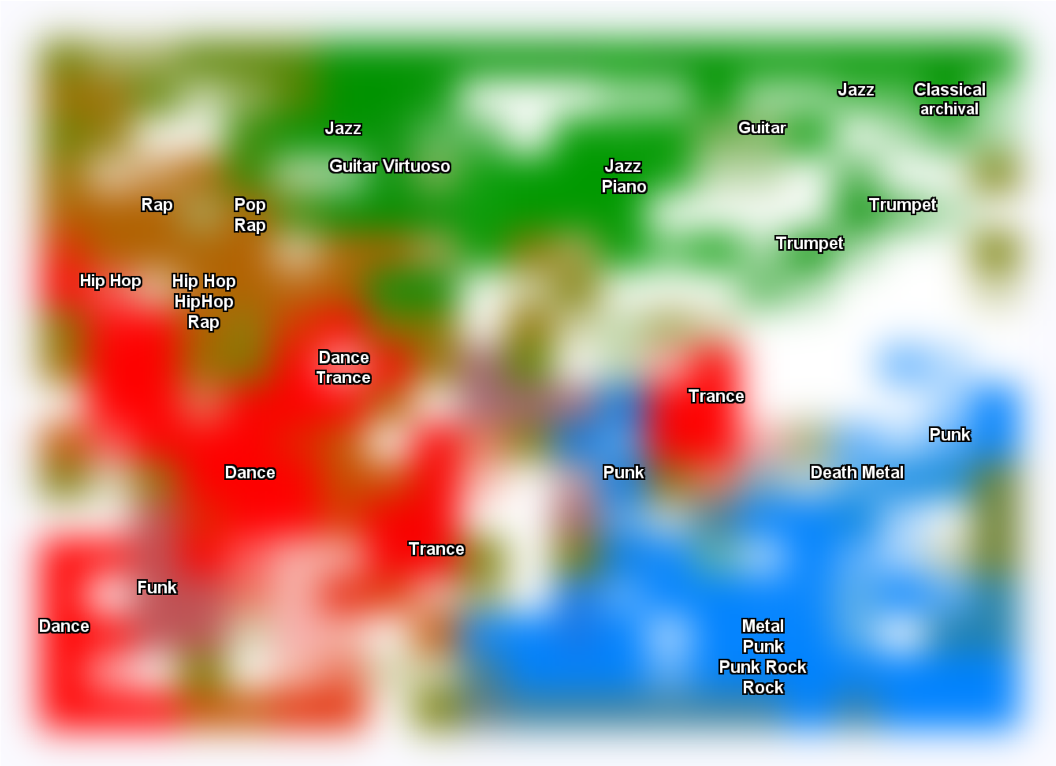


Figure 4.5: Screenshot of a Colour Music Description Map prototype. In addition to labels, similar colours indicate similar music also on disconnected regions on the map. Here, for instance, red areas indicate electronic dance music. An enclave of Trance music can be spotted easily.

4.3 Augmented Exploration in Virtual Music Landscapes

The potential applicability of the MDM in interactive music browsing interfaces has already been addressed in the last section. This section deals with the integration of the MDM technique into the nepTune interface, an innovative three-dimensional user interface for exploring music collections (cf. [Knees et al., 2006b,Knees et al., 2007b]).

The intention behind nepTune is to provide an interface to music collections that goes beyond the concepts of conventional User Interfaces (UIs), and hence refrain from including the components contained in almost every window toolkit. Instead, it should appear like a video game, making it easy and attractive to use. Built upon the "Islands of Music" metaphor (see Section 3.3.2), an island landscape is generated where similar sounding pieces are grouped together and accumulations of similar music are represented as mountains. Figure 4.6 shows a view on such a generated music landscape. Within the terrain created, the user can move freely to explore the contained music collection and also hear the closest sounds with respect to his/her current position. The aim of integrating the MDM is to automatically add descriptors of the music that serve as landmarks and thus augment the process of interactive exploration, as elaborated in Section 4.3.2. Before that, in the next section, a description of the nepTune system is given.

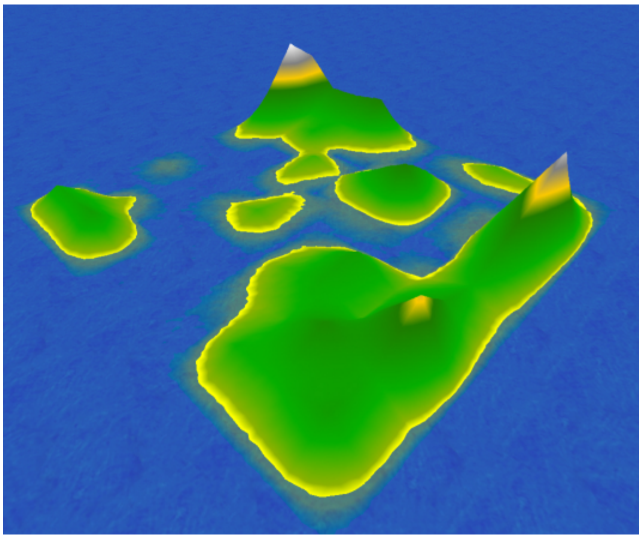


Figure 4.6: A view over a nepTune-created music landscape. Exploration of the underlying collection is enabled by freely navigating through the landscape and hearing the music typical for the region around the listener's current position.

4.3.1 Prototype: The nepTune Interface

The general philosophy underlying nepTune is that music collections should be structured (automatically, by the computer) and presented according to intuitive musical criteria, and that music interfaces should permit and encourage the creative exploration of music repositories, and new ways of discovering hidden treasures in large collections. To make this kind of philosophy more popular, the first targeted application will be as an interactive exhibit in a modern science museum. Thus, the main focus was not on the applicability as a product ready to use at home. However, that could be achieved with little effort by incorporating standard music player functionalities. Before describing the technical details of the interface, the conceptual background of nepTune is discussed.[4]

Interface Concept

As a central aspect of the interface, similar sounding pieces are automatically grouped together. Thus, the more similar pieces the user owns, the higher is the terrain in the corresponding region. To achieve this, essentially, a music map is created and transformed to an artificial, but nevertheless appealing three-dimensional island landscape (cf. Figures 4.7 and 4.8). Each music collection creates a characteristic and unique landscape. Another important aspect of the interface is the fact that the music surrounding the listener is played during navigation. Hence, it is not necessary to select each song manually and scan it for interesting parts. While the user explores the collection he/she is automatically presented with audio thumbnails from the closest music pieces (i.e., the central 30 seconds of each piece), giving immediate auditory feedback on the style of music in the current region. Thus, the meaningfulness of

[4] For implementation details, such as underlying software frameworks and libraries, the reader is referred to the original publications, i.e., [Knees et al., 2006b, Knees et al., 2007b].

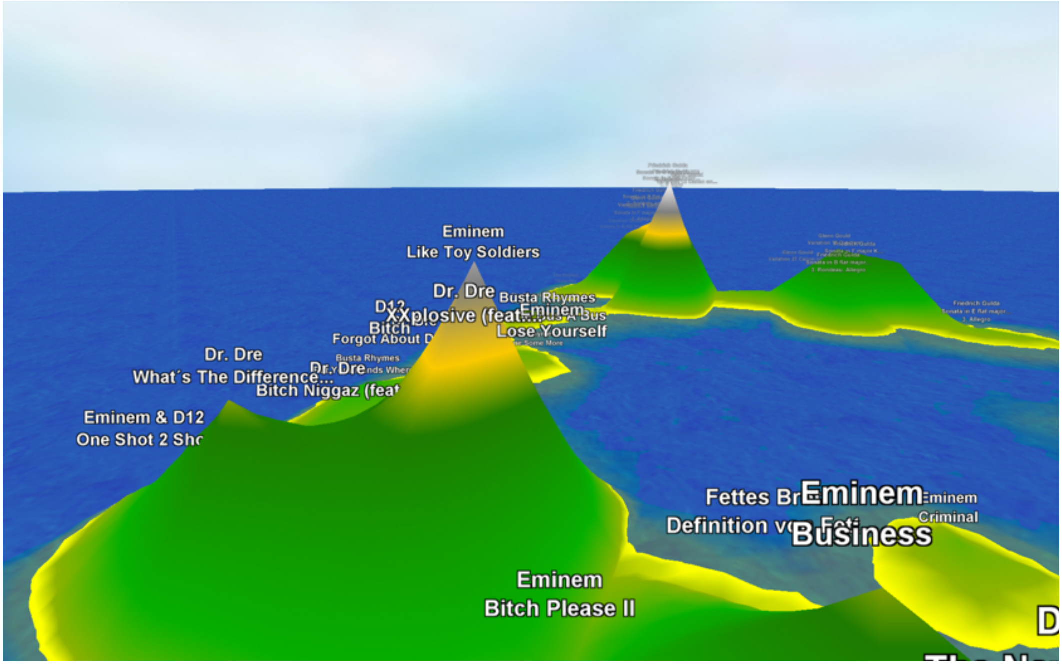


Figure 4.7: Screenshot from the nepTune interface showing a group of Rap music islands in the front and two connected islands containing piano music in the back.

the spatial distribution of music pieces in the virtual landscape can be experienced directly. As mentioned, the main intention of nepTune is to provide an interface to music collections that goes beyond the conventional computer interaction metaphors and that should be fun to use and engage people. Therefore, rather than constructing an interface that relies on the classical point-and-click scheme best controlled through a mouse, the whole application is designed to be controllable with a standard game pad as used for video game controlling. A game pad is well suited for exploration of the landscape as it provides the necessary functionality to navigate in three dimensions whilst being easy to handle. Furthermore, the closeness to computer games is absolutely intended to emphasize the “fun factor”.

In the envisioned scenario, i.e., as an interactive exhibit, visitors are encouraged to bring their own collection, e.g., on a portable mp3 player, and are given the opportunity to explore their collection like a landscape. In its current implementation, the process is invoked by the user through connecting his/her portable music player via an USB port. This is automatically recognised by nepTune, and the system then randomly extracts a pre-defined number of audio files from the player and starts to extract audio features from these (see below). A special challenge for applications that are presented in a public space is to perform computationally expensive tasks like audio feature analysis while keeping visitors motivated and convincing them that there is actually something happening. Hence, to indicate the progress of audio analysis, it is visualised via a little animation: small, coloured cubes display the number of items left to process. For each track, a cube with the number of the track pops up in the sky. When the processing of an audio track is finished, the corresponding cube drops down and splashes into the sea. After all tracks have been processed (and after a clustering and structuring phase that is hidden from the

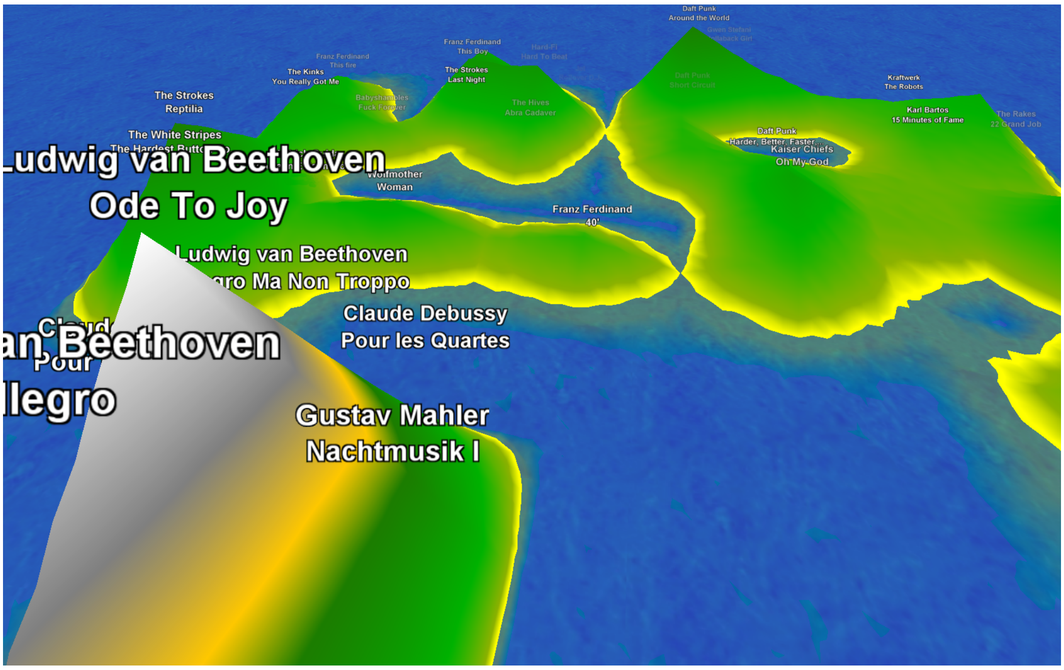


Figure 4.8: Another screenshot from the nepTune interface. The large peaky mountain in the front contains classical music. The classical pieces are clearly separated from the other musical styles on the landscape. The island in the left background contains Alternative Rock, while the islands on the right contain electronic music.

user), an island landscape that contains the tracks emerges from the sea. Then, it is the user's turn to explore the collection by means of a game pad.

The three-dimensional landscape is displayed in front of the user. As described before, while moving through the terrain, the closest sounds with respect to the listener's current position can be heard from the directions where the pieces are located, to emphasize the immersion. Thus, in addition to the visual grouping of pieces conveyed by the islands metaphor, islands can also be perceived in an auditory manner, since one can hear typical sound characteristics for different regions. For optimal sensation of these effects, sounds are output via a 5.1 surround audio system. Detaching the USB storage device (i.e., the mp3 player) causes all tracks on the landscape to immediately stop playback. The game pad is disabled and the viewer's position is moved back to the start. Subsequently, the landscape sinks back into the sea, giving the next user the opportunity to explore his/her collection.

Technical Realisation

In principle, creation of the islands terrain utilises the same techniques as Pampalk's Islands of Music approach (see Section 3.3.2). A difference consists in the audio extraction step. For the version created to serve as a public installation, the number of files to process is limited to 50 mainly for time reasons, since the application should be accessible to many users. From the chosen audio files, the central 30 seconds are extracted and analysed. These 30 seconds also serve as looped audio thumbnail for spatialised playback in the landscape. The idea is to extract the audio features only from a typical section of the track which is usually found in the central part rather than at beginning or end.

As with the Islands of Music, an SDH is calculated to create a three-dimensional landscape model that contains the musical pieces. However, in the plain SOM representation, the pieces are only assigned to a cluster rather than to a precise position. Thus, a strategy to place the pieces on the landscape has to be developed. The simplest approach would be to spread them randomly in the region of their corresponding map unit. That has two drawbacks: the first is the overlap of labels, which occurs particularly often for pieces with long names and results in cluttered maps. The second drawback is the loss of similarity information inherent in the ordering of pieces based on their distance to the unit's centre. It is desirable to have placements on the map that reflect the positions in feature space in some way.

The solution finally adopted consists in defining a minimum distance d_m between the pieces and placing the pieces on concentric circles around the map unit's centre such that this distance is always guaranteed. To preserve at least some of the similarity information from feature space, all pieces are sorted according to their distance to the model vector of their best matching unit in feature space. The first item is placed in the centre of the map unit. Then, on the first surrounding circle (which has a radius of d_m), at most ($2\pi \approx 6$) can be placed such that d_m is maintained (because the circle has a perimeter of $2d_m\pi$). The next circle (radius $2d_m$) can host up to ($4\pi \approx 12$) pieces, and so on. For map units with few items, the circle radii are scaled up, to distribute the pieces as far as possible within the unit's boundaries. As a result, the pieces most similar to the cluster centres are kept in the centres of their map units and also distances are preserved to some extent. More complex (and computationally demanding) strategies are conceivable, but this simple approach works well enough for the given scenario.

To allow for a more focused browsing, when moving through the landscape labels of pieces far away from the listener are faded out. This is accomplished by utilising the *mipmap* functionality of the used rendering engine (see [Williams, 1983]). Hence, only the pieces close to the current position are clearly visible, while more distant pieces are just indicated. To still allow for an overview over the music contained in other regions, the MDM is incorporated as described in the next section.

4.3.2 Incorporating Web-Based Labels and Images for Orientation

In addition to the auditory feedback that supports the user in exploring the collection, meaningful term descriptors of the music in the specific regions that serve as landmarks and facilitate orientation can be very useful. Since both nepTune and MDM build upon the idea of music maps, integrating the MDM into nepTune to enable this additional feature is a straightforward task. However, after creating the MDM, placement of the terms to be displayed on the landscape is non trivial. For the MDM, relevance of terms to map units can be easily grasped since they are displayed either directly on the grid's corresponding cell or, in the case of connected clusters, within their larger boundaries, since separating grid markers are removed in order to indicate the coherence of similar cells (cf. Figure 4.3). On the nepTune landscape, the structure of the underlying SOM should be concealed from the user. Thus, no grid lines or other markers indicate the boundaries of the clusters. Display-

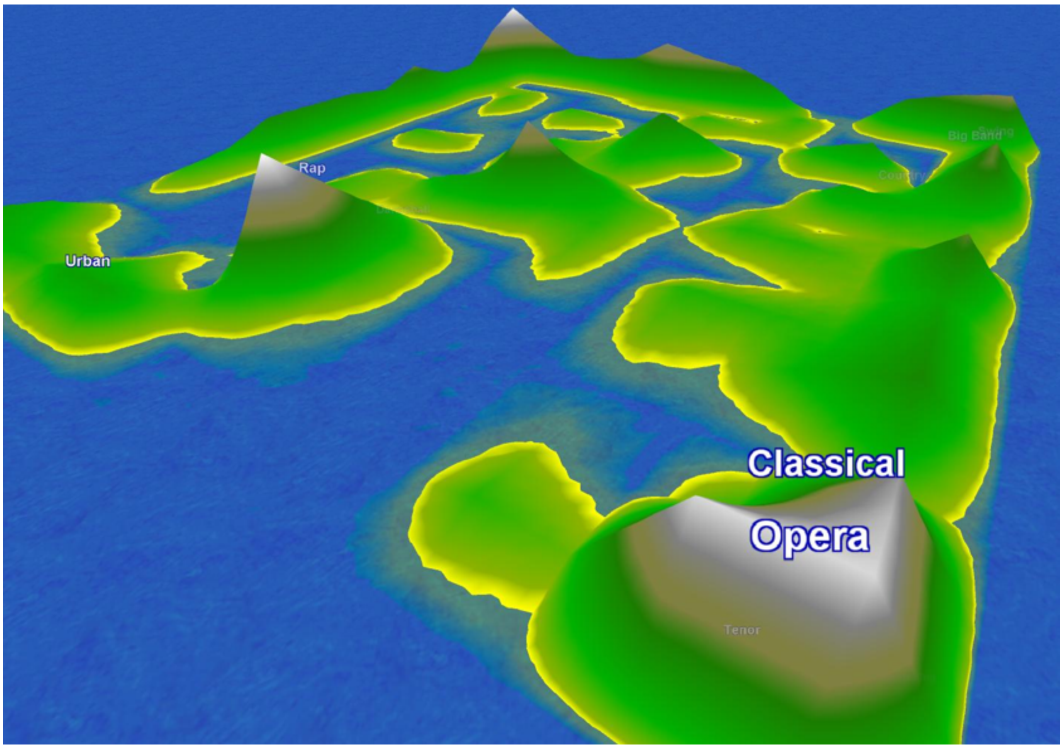


Figure 4.9: nepTune with integrated MDM: Important terms serve as visible landmarks.

ing terms on the landscape such that they can serve as landmarks for orientation therefore requires a strategy that positions terms in an intuitive fashion.

To find suited positions for the descriptors on the map, the following heuristic is applied: First, the centre of each description cluster has to be determined. To this end, the centre of gravity of all contained map units of a connected cluster is calculated. In case the centre of gravity lies outside the area of any of the connected map units, the centre of the closest contained map unit is chosen instead. For terms with a term-goodness score $G > 0.33$ (cf. Section 4.2.2), the same positioning strategy as for the music pieces is applied, i.e., placement of the term labels on concentric circles around the cluster's centre. Starting with placing the label of the highest scoring term in the centre, labels are assigned to surrounding circles according to their ranking of G values. For terms with a term-goodness score $G \leq 0.33$, corresponding labels are distributed randomly within the corresponding map unit cluster.

Regarding the presentation of the labels, the same technique as on the MDM, i.e., selection of the label's font size depending on the score of the term, is again applied here. Hence, terms that are very important for specific clusters have better visibility in the landscape. Together with the mipmap functionality also applied to music piece labels, i.e., fading out labels as they become smaller, this results in a few landmarks visible over a long distance (see Figure 4.9) that support a global orientation. Other, more specific labels only become apparent when the corresponding regions are explored (see Figure 4.10).

In addition to displaying related terms to allow for an augmented exploration of the nepTune landscape, exploration can be enriched with an alternative mode that

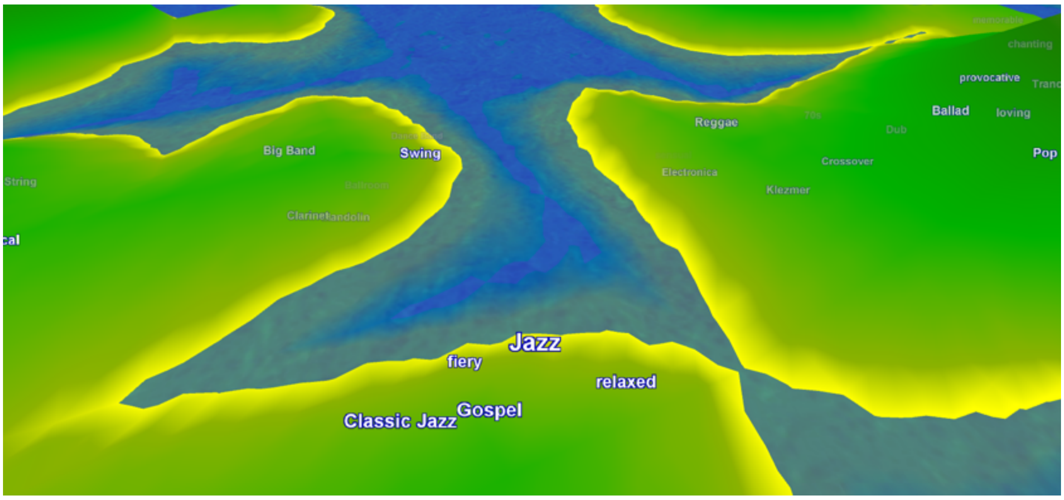


Figure 4.10: nepTune showing specific terms describing the surrounding music.

adds extra visual elements, more precisely images from the Web that are related to the artists and the nouns on the MDM. To obtain related images, the images search function of Google is invoked. For each track, a query consisting of the corresponding artist's name is sent to Google and one of the three highest ranked images is then displayed instead of the track's meta-data on the landscape. For each noun appearing on the MDM-enhanced landscape, a related image is obtained by sending the noun itself as query and randomly selecting one of the first ten retrieved images to be displayed instead of the noun. Adjectives appearing on the landscape are ignored since searching for adjectives in an image search results mostly in inappropriate results, especially in the music context. Figure 4.11 depicts the image-enhanced version of the scenery contained in Figure 4.10. It should be noted that no further filtering mechanisms that aim at identifying "correct" visual content are applied. The intention is just to add another dimension of fun and enjoyment by incorporating images from "the Web" irrespective of whether these are relevant from a traditional IR perspective or completely unrelated to the contained music. Hence, for *Rock* music images of rocks may appear as well as pictures of houses for *House* music.

After incorporating the MDM and the mode containing related images, nepTune now provides a total of four different modes to explore the landscape, namely a plain landscape mode without any labels, the default mode that displays artist name and song name as given by the ID3 tags of the mp3 files, the MDM mode with typical words that describe the heard music, and a mode where images from the Web are presented that are related to the artists and the descriptions. For comparison, screenshots of these four modes can be seen in Figure 4.12.

4.3.3 Extensions and Future Directions

In its current state, nepTune has a focus on interactive exploration rather than on providing full functionality to replace existing music players. However, recently extensions that allow to use nepTune to access, for instance, private collections at home have been made. A central step in this direction is allowing the user to select

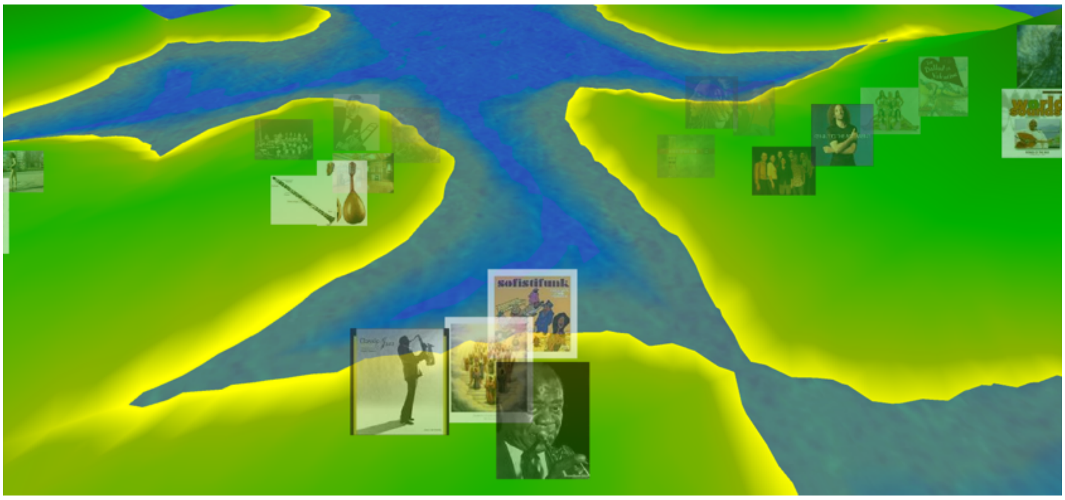


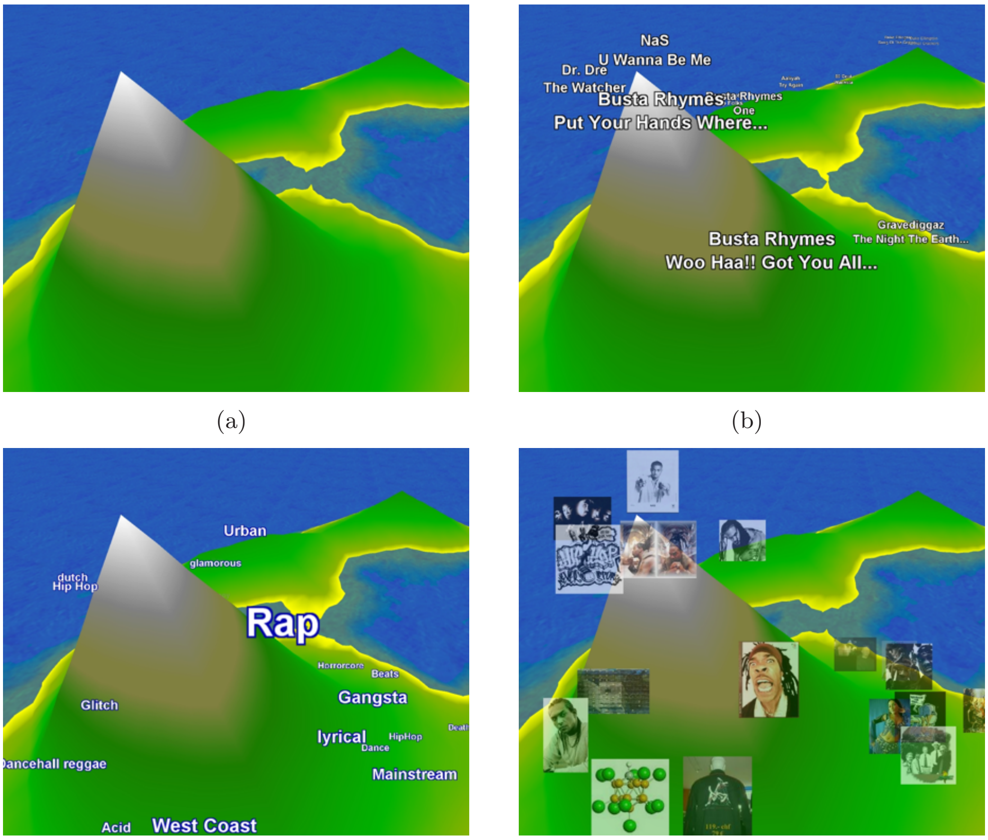
Figure 4.11: nepTune showing images based on artists and related nouns (cf. Figure 4.10).

specific tracks. Currently, this is accomplished by putting the focus on the closest track in front of the listener's position. Pressing a button on the game pad then confirms the selection. This enables focused listening, i.e., listening to a specific track from beginning to end, and would give the opportunity to present additional track-specific meta-data for the currently selected track, i.e., as in other music player applications, further ID3 tags like album or track length, as well as lyrics or the correct album covers could be displayed. Furthermore, by enabling selection of specific tracks, the application can be easily extended to provide such useful methods as automatic playlist generation, as has been done recently by [Krenmair, 2008]. This extension allows to determine a start and an end song on the map and then finds a path along the distributed pieces on the map that serves as playlist.[5] Such a path can be easily visualised on the landscape and provide some sort of "auto-pilot mode", where the movement through the landscape is done automatically by following the playlist path (Figure 4.13).

The biggest challenges are presented by larger collections (containing tens of thousands of tracks). One option could consist in incorporating a hierarchical extension to the SOM such as the GHSOM (e.g., [Rauber et al., 2002]). Another option could be a *level-of-detail* approach that makes use of the music descriptors extracted from the Web. At the top-most level, i.e., the highest elevation, only broad descriptors like musical styles would be displayed. Reducing the altitude would switch to the next level of detail, making more distinct descriptors appear, along with very important artists for that specific region. Single tracks could then be found at the most detailed level. Thus, the relatedness of the interface to geographical maps would be emphasised and the application would act even more as a "flight simulator for music landscapes".

Another future application scenario concerns mobile devices. Currently, versions of the nepTune interface for Apple's iOS platform and Google's Android platform are being developed. While the audio feature extraction step still should be carried

[5] A similar idea to create playlists by selecting start and end songs has later been published by [Flexer et al., 2008].



(a) (b) (c) (d)

Figure 4.12: Four screenshots from the same scene in nepTune's four different exploration modes. (a) depicts the plain landscape without any labels. (b) shows the default mode, where artist and song name are displayed. Since the island contains Rap music, tracks of artists like *Busta Rhymes* and *NaS* can be found. (c) shows the MDM mode with typical words that describe the music, such as *Rap*, *Gangsta*, *West Coast*, *lyrical*, or *Mainstream*. (d) depicts the mode in which related images from the Web are presented on the landscape. In this case, these images show Rap artists (*Busta Rhymes, 2Pac, etc.*), as well as related artwork.

out on a personal computer for reasons of performance and battery runtime, it is possible to perform the remaining steps, i.e., training of the SOM and creation of the landscape, on the mobile device. Considering the ongoing trend toward mobile music applications and the necessity of simple interfaces to music collections, the nepTune interface could be a useful and fun-to-use approach to accessing music also on portable devices.

4.4 Evaluation

This section deals with evaluating the MDM technique for labelling music maps as well as the nepTune interface. In both cases, a small group of humans has been drawn on to assess the qualities of the approaches. In the case of the MDM, a quantitative evaluation has been carried out, for the nepTune interface, participants have been asked to report on their impressions when using the system.

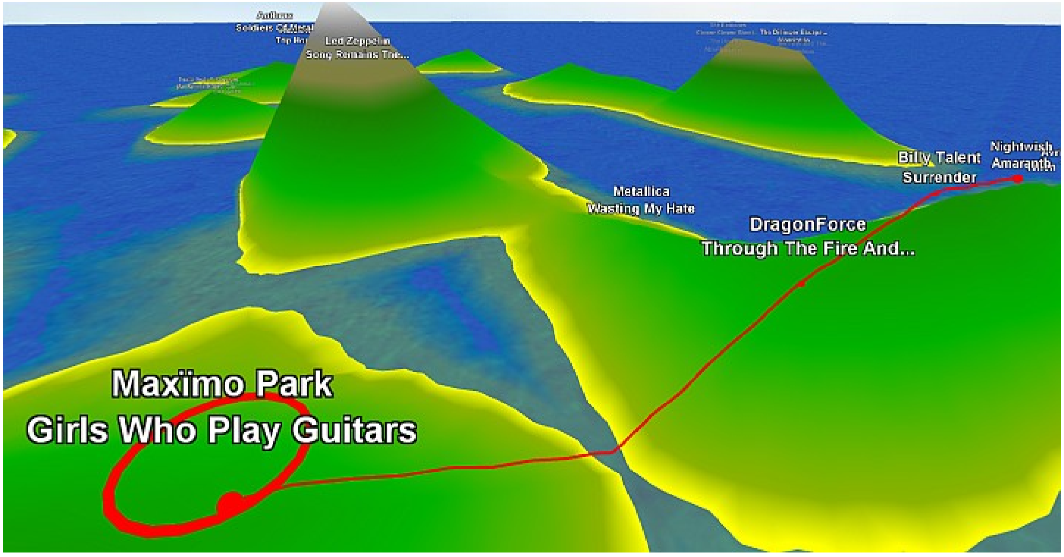


Figure 4.13: Using nepTune for automatic playlist generation

4.4.1 User Evaluation of the Music Description Map

Evaluating the MDM quantitatively is non trivial. The main reason is the lack of any form of ground truth, i.e., a corpus of music pieces specifically pre-labelled with the used Web vocabulary. Furthermore, it is not the primary intention of the evaluation to assess the quality of the retrieved Web data with respect to a given ground truth, but rather to see whether the derived cluster descriptors can assist users in browsing music collections. More specifically, it is of interest to investigate if users can predict which music they will find in which regions, i.e., if the region-specific descriptors are useful. Hence, six users were asked to provide a small selection of music pieces from their personal collection, i.e., music they are familiar with. Based on every collection, a small MDM (6×4) was created, which was presented to the corresponding user together with a list of the contained music pieces. The users were then asked to assign each track to the cluster that best describes each track in their opinion. In case of uncertainty, it was also possible to select a second best matching cluster. The results of this evaluation can be found in Table 4.1. For each participant, the collection size (i.e., how many tracks were made available for the study by each participant), the number of emerging description clusters, and the number of tracks correctly located (at first try, at second try, or in total) are given. The total number of matching assignments is also related to the collection size and expressed in terms of percentage.

Obviously, the results are very heterogeneous. At first glance, a high number of emerging clusters seems to be responsible for poor results. A deeper investigation reveals that both, high number of clusters and bad results, have the same sources, namely many non-English music pieces and many outliers, i.e., single pieces that are placed on the map away from the others and therefore form their own distinct description clusters. In test case 6, the collection basically consisted solely of Rap and Dance music with strong beats and all clusters on the map were labelled very similarly. In contrast, collections that contained consistent subsets of music (which

Test Person	**1**	**2**	**3**	**4**	**5**	**6**	**total**
Tracks in Collection	54	35	28	45	51	41	254
Clusters on MDM	8	5	13	8	6	12	
Matching Assignments (1^{st} Choice)	21	18	7	10	42	7	105
Matching Assignments (2^{nd} Choice)	5	4	n.a.	6	3	0	18
Total Matching Assignments	26	22	7	16	45	7	123
Total Matching Assignments (%)	48.1	62.9	25.0	35.6	88.2	17.1	48.4

Table 4.1: Evaluation results of track to MDM cluster assignment.

could be identified by the audio measure) led to few large, clearly separated clusters. On these collections, highest matchings between MDM and user opinions could be observed. The conclusion drawn from this evaluation is that the quality and usefulness of descriptions relies heavily on the consistency of the cluster contents and the separability of the music collection. Since the capabilities of the underlying similarity measure are key factors to both of these aspects, the sensitivity of the approach to the used audio-based features becomes apparent.

4.4.2 Qualitative Evaluation of the nepTune Interface

To gain insights into the usability of nepTune, a small user study has been conducted. Eight persons were asked to play with the interface and to report on their impressions afterwards. In general, responses were very positive. People reported that they enjoyed the possibility to explore and listen to a music collection by cruising through a landscape. While the option to display related images on the landscape was generally considered to be a "nice-to-have" extension, the option to display related words was rated to be a valuable add-on, even if some of the displayed words were confusing for some users. Controlling the application with a gamepad was intuitive for all users.

Sceptical feedback was mainly caused by music auralisation in areas where different styles collide. However, in general, auralisation was rated positively, especially in regions containing *Electronic Dance Music*, *Rap/HipHop*, or *Classical Music*, since it assists in quickly identifying groups of tracks from the same musical style. Two users suggested creating larger landscapes to allow more focused listening to certain tracks in crowded regions — a feature that has been enabled in nepTune in the meantime by allowing to select specific tracks (cf. Section 4.3.3).

4.5 Recapitulation and Discussion

This chapter has dealt with the application of meaningful descriptors to music pieces arranged on a music map. The descriptors are automatically extracted from the Web and assigned to the pieces using a top-down technique called Music Description Map (MDM). Furthermore, the MDM technique has been incorporated into the nepTune music exploration interface to allow for better orientation when navigating through the virtual landscape.

As can be seen in Figure 4.3, in most cases, terms describing the style and genre of music are typically most important (i.e., most discriminative) to describe the content of a cluster and to aid the user in finding music. However, experiments with users showed that there is still ample space for improvements. While well known music is substantially represented on the Web and can also be sufficiently captured by the used vocabulary, many non-English music styles can not be described and result in misleading terms. Furthermore, acoustic outliers that are not clustered with other pieces also impose some problems on the labelling. In fact, the MDM assumes a “perfect” underlying similarity measure and clustering, meaning that clusters should generally contain consistent music that can be described by the used vocabulary and that no outliers should be placed on the map (although it is clear that this will never be satisfied in practice, it should be stated that unexpected descriptions on the map are frequently caused by misplaced music pieces).

Since a hybrid technique like the MDM operates on the combination of content- and context-based music information, a mismatch between these two complementary sources, leading to surprising results, is very likely. Hence, one can not expect to get perfect descriptions all the time. However, it has been shown that the presented technique is capable of incorporating contextual information sufficiently well to support users in finding music. Especially in the context of the nepTune interface, which puts an emphasis on the entertaining aspects of music browsing and does not require a perfectly labelled landscape, the MDM has proven useful. In summary, integrating the MDM technique has extended nepTune to a multimedia application that examines several aspects of music and incorporates information at different levels of music perception — from the pure audio signal to culturally determined meta-descriptions — which offers the opportunity to discover new aspects of music. Not least due to the MDM extension, this makes nepTune an interesting medium for exploring music collections.

Chapter 5

Larger Scale Indexing and Retrieval of Music via Web-Data

While the focus so far was on deriving music-specific terms from the Web mainly for labelling of pieces to support the process of browsing, this chapter presents approaches to automatically build a search engine for music collections that can be queried through natural language. The goal is to perform music piece indexing also on a larger scale and eliminate the necessity of pre-clustering a music collection based on audio features.

Most existing (purely text-based) approaches to music search engines depend on explicit manual annotations and meta-data assigned to the individual audio pieces. In contrast, here, based on the ID3 tags of a collection of mp3 files, relevant Web pages are retrieved via a Web search engine and the contents of these pages are used to characterize the music pieces. This allows for unsupervised indexing using a rich vocabulary. Conceptually, this approach is comparable to Web image indexing approaches that make use of contextual text information (see Section 2.2.2). However, the difference is that in this case the indexed music pieces are not necessarily located on the Web, and therefore a direct (local) context is not available. Thus, the context has to be "approximated" by exploiting Web pages that contain references to the tracks' meta-data. In the following, various methods and possible extensions are investigated, namely

- two different indexing (and retrieval) techniques,
- techniques for filtering of noisy data, and
- a technique to incorporate audio-based similarity information.

A schematic overview of the techniques elaborated in this chapter can be found in Figure 5.1.

5.1 Motivation

Over the past years, using text-based search engines has become the "natural" way to find and access all types of multimedia content. While there exist approaches to automatically derive and assign "semantic", natural language descriptors for images, videos, and – of course – text, the broad field of (popular) music has not drawn

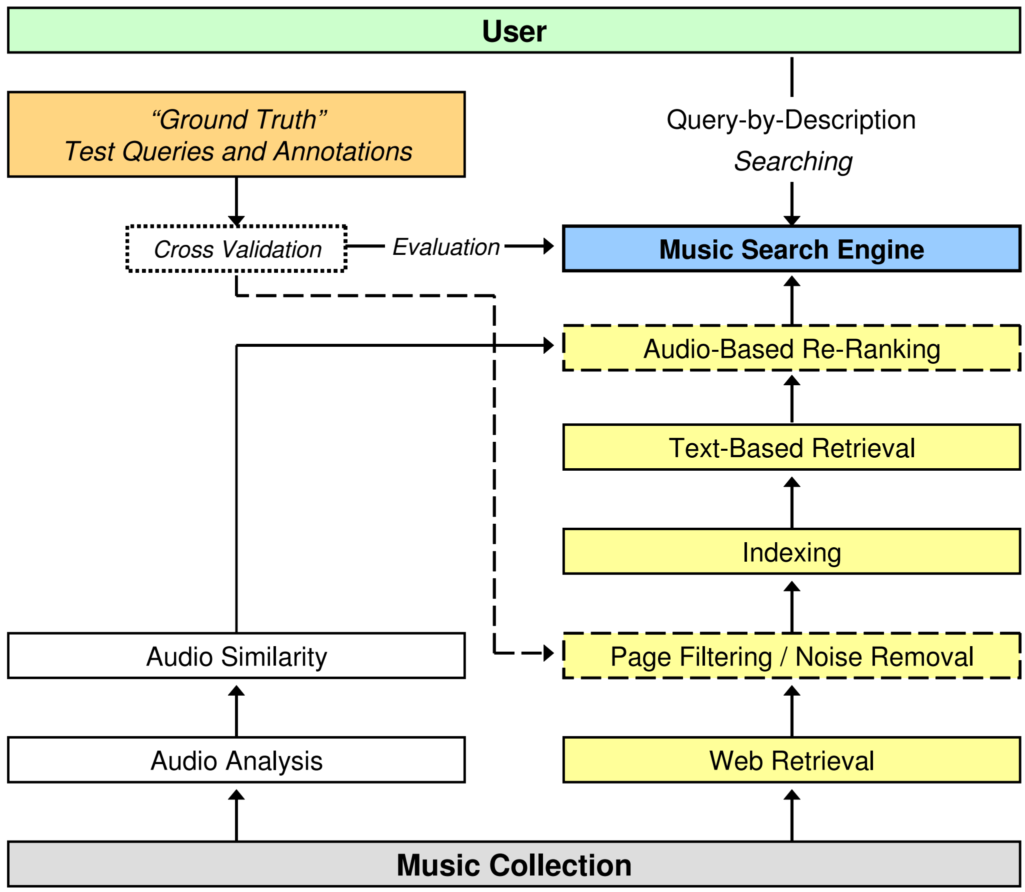


Figure 5.1: Schematic overview of the methods presented in this chapter. Yellow bars with solid border indicate the steps proposed to build a text-based music search engine; dashed borders indicate optional steps also presented in this chapter, i.e., page filtering prior to indexing (unsupervised as well as supervised by exploiting the ground truth information represented by the red box) and re-ranking of results based on audio similarity.

that much of attention. Basically all existing music search systems, e.g., those offered by commercial music resellers, make use of manually assigned subjective meta-information like *genre* or *style* (in addition to more or less objective meta-data categories like *artist name*, *album name*, *track name*, or *year of release*) to index the underlying music collection. A person interested in finding music, e.g., a potential customer, must have a very precise idea of the expected results already before issuing the query. The intrinsic problem of these indexing methods is the limitation to a rather small set of meta-data, neither capable of capturing musical content nor cultural context of music pieces.

However, as digital catalogues rapidly become larger and more inconvenient and inefficient to access, the need for more sophisticated methods that enable intuitive searching inside large music collections increases. For example, instead of just retrieving tracks labelled as *Rock* by some authority, a system should allow for formulating a query like *"Rock with Great Riffs"* to find songs with energetic

	rock with great riffs		*dreamy*
1.	**Chuck Berry - Roll over Beethoven**	1.	**Psychetropic - Dead slow day**
2.	**Def Leppard - Pour some sugar on me**	2.	My Bloody Valentine - When you sleep
3.	**Pantera - Becoming**	3.	**Mazzy Star - Fade into you**
4.	**Buffalo Springfield - Mr. Soul**	4.	T. Rex - Children of the revolution
5.	Bob Seger - Turn the page	5.	**Kourosh Zolani - Peaceful planet**
6.	**Thin Lizzy - Don't believe a word**	6.	Stereolab - Cybeles reverie
7.	**Seismic Anamoly - Wreckinball**	7.	Altered Images - Don't talk to me about...
8.	**Screaming Trees - Nearly lost you**	8.	**Aerobic Jonquil - Sweat machine**
9.	**Lynyrd Skynyrd - Sweet Home Alabama**	9.	**Michael Masley - Advice from the angel...**
10.	**Queen - We will rock you**	10.	Catherine Wheel - Black metallic
11.	**Led Zeppelin - Immigrant song**	11.	Cat Power - He war
12.	**AC/DC - Dirty deeds done dirt cheap**	12.	Air - Sexy boy
13.	**Carl Perkins - Matchbox**	13.	**Myles Cochran - Getting stronger**
14.	White Stripes - Hotel Yorba	14.	Kenji Williams - I'm alive
15.	**Urge Overkill - Sister Havana**	15.	Yo la tengo - Tom Courtenay
16.	**Darkness - I believe in a thing called love**	16.	**Shins - New slang**
17.	**Steely Dan - Rikki don't lose that number**	17.	Cure - Just like heaven
18.	**Kiss - Deuce**	18.	Alicia Keys - Fallin'
19.	**Cheap Trick - Dream Police**	19.	**Stan Getz - Corcovado quiet nights...**
20.	**Soundgarden - Black hole sun**	20.	**Spiritualized - Stop your crying**
21.	Standells - Dirty water	21.	**Kraftwerk - Spacelab**
22.	**Black Sabbath - Black Sabbath**	22.	Association - Windy
23.	**Byrds - Wasn't born to follow**	23.	Seismic Anamoly - Wreckinball
24.	**Black Crowes - Thorn in my pride**	24.	**Williamson - What's on the ceiling...**
25.	**Sex Pistols - Pretty vacant**	25.	Arthur Yoria - At least you've been told
26.	**Troggs - Wild thing**	26.	Sundays - Here's where the story ends
27.	**Smashing Pumpkins - Rocket**	27.	Sebadoh - Soul and fire
28.	**Boston - More than a feeling**	28.	Emma's Mini - Lost
29.	**Steppenwolf - Born to be wild**	29.	Cranberries - Linger
30.	Strawbs - New world	30.	Manassas - Bound to fall

Table 5.1: Exemplary rankings for the queries *rock with great riffs* and *dreamy* as returned by one of the retrieval approaches presented in this chapter. Bold entries indicate relevant results according to the author of this thesis.

guitar phrases. Table 5.1 shows an exemplary ranking obtained for that query to demonstrate the potential of the methods presented in this chapter. Clearly, music resellers with very large music databases or music information systems could benefit from such an engine, as it provides access to their catalogue in the most common and most accepted manner.

The presented methodology builds upon the work by [Knees et al., 2007a], who present first steps towards the task of building a search system capable of satisfying arbitrary natural language queries. In this first approach, for each track in a collection of mp3 files, a set of relevant Web pages is retrieved via Google. This textual data allows to represent music pieces in a traditional term vector space (cf. Section 3.1.3). Additionally, this contextual information is combined with information about the content by incorporating an audio similarity measure, which allows for reduction of the dimensionality of the feature space, as well as description of music pieces with no (or little) related information present on the Web. This method is also applied in Chapter 6 where it provides the foundation for the adaptive retrieval method proposed. Hence, for details about this first approach, the reader is referred to Chapter 6 and [Knees et al., 2007a]. Throughout this chapter, interesting differences to the initial approach are pointed out.

In this chapter, the research initiated in [Knees et al., 2007a] is substantiated and further steps towards a textually driven music search engine are presented. The idea of collecting Web pages via a Web search engine as a data source also pertains to the methods presented here. Building upon that type of data, two different indexing and corresponding retrieval strategies are elaborated (Section 5.3). Furthermore, the effects of noisy data detection and filtering as well as the potential of audio-based ranking modifications are investigated (sections 5.4 and 5.5, respectively). For extensive evaluation, two test collections with different characteristics as well as two different Web search engines are employed.

5.2 Web-Data Acquisition

The idea of Web-based indexing is to collect a large number of texts related to the pieces in the music collection to gather many diverse descriptions (and hence a rich indexing vocabulary) and allow for a large number of possible queries. As in Chapter 4, where overcoming the limitation of Web-MIR approaches to the artist level forms one of the goals, also for this task it is an objective to derive descriptors for individual tracks. In contrast to Chapter 4, this is not accomplished by finding discriminative terms based on an audio similarity clustering, but by obtaining more specific data from the Web. However, typically, when trying to obtain track-specific Web pages, the number of available pages varies considerably depending on the track in question (cf. [Knees et al., 2007a]). To gather information that is very specific for a track but also to gather a high number of available Web pages (via artist related pages), the results of three queries that are issued to a Web search engine are joined for each track m in the collection M:

1. "*artist*" music
2. "*artist*" "*album*" music review -lyrics
3. "*artist*" "*title*" music review -lyrics

While the first query is intended to provide a stable basis of artist related documents, the second and third query target more specific pages (reviews of the album and the track, respectively). The additional constraints to the queries have been chosen based on experiences in related work, cf. [Knees et al., 2008b]. Since searching for album or track names typically yields a majority of results containing corresponding lyrics (which, in general, do not consist of the desired type of descriptors), the *-lyrics* constraint is added to exclude as many of these pages in advance as possible.[1] For each query, at most 100 of the top-ranked Web pages are downloaded. The set of pages associated with music track m is in the following denoted as D_m and consists of all Web pages retrieved via the queries described. That is, in case of completely distinct result sets for the three queries, a total of 300 Web pages is assigned to each track. Furthermore, all retrieved documents are also stored in an index $I = \bigcup_{m \in M} D_m$.

[1] Unfortunately, it is not feasible to completely filter all lyrics pages by adding constraints to the query. Additionally, it has to be kept in mind that there is a trade-off between filtering all unwanted material and unintentional removal of relevant information, e.g., on review pages that contain links to lyrics.

In practice, the Java-based open source search engine Nutch is applied to perform the retrieval of all Web pages and to store them in a Lucene index. This allows for efficient access to the retrieved documents in the following steps (cf. Section 3.1).

5.3 Indexing and Retrieval

After obtaining and indexing the related Web pages, the next step consists in applying methods to associate the textual data with the music pieces (music piece indexing) and to develop methods that enable music piece retrieval based on these indexing strategies. As mentioned, the idea is to collect a substantial amount of texts related to the pieces in the music collection to obtain diverse descriptions and a rich indexing vocabulary that allows for a large number of possible queries. In the initial approach in [Knees et al., 2007a], virtually any query is permitted by involving Google and downloading the top results for automatic query expansion, i.e., by finding additional, related terms to broaden the result set.[2] By using the standard retrieval approach included in Nutch (see sections 3.1.2 and 3.1.3), the methods presented here restrict the indexing vocabulary to terms that are included in the retrieved Web data. This is not necessarily a drawback, since this still comprehensive vocabulary is limited to terms that actually occur in the context of the music pieces. The reasons for the decision to waive on-line query expansion are manifold. First, the dependency on Google to process queries is a severe drawback. Besides the fact that automatically querying Google is limited to 1 000 queries per day, it is very uncommon to need access to the Internet to query a local database. Furthermore, response time of the system increases by the time necessary to perform the on-line search and download the Web pages.

Therefore, the two methods presented in the following only build upon the data obtained in the initial data acquisition step. The first method is conceptually very similar to the method presented in [Knees et al., 2007a], but uses the Lucene scoring scheme for retrieval. The second method pursues a different scoring approach centred around relevance ranking of Web pages based on the Nutch index.

5.3.1 Pseudo Document Vector Space Model Approach

The basic idea behind this approach to associating Web-data to individual music pieces is simply to agglomerate all information found and treat this agglomeration as one document. More precisely, for each music piece m, all retrieved texts (i.e., all texts $d \in D_m$) are concatenated into a single *pseudo document* ψ_m. All resulting pseudo documents are then indexed (see Section 3.1.2). Hence, each music piece

[2] In [Knees et al., 2007a], a query to the system is processed by adding the constraint *music* and sending it to Google. From the 10 top-ranked Web pages, the "expanded" query vector is constructed, i.e., a TF·IDF vector in the same feature space as the term vectors for the tracks in the music collection. This less sparsely populated query vector is then compared to the music pieces in the collection by calculating cosine distances between the respective vectors. Based on the distances, a relevance ranking is obtained. Since no thresholding or other constraints are applied for ranking, in [Knees et al., 2007a], for a query, the whole collection is sorted according to relevance. In contrast, for the methods presented here, in a first step only tracks that contain any of the query terms are selected. After that, only this subset of the collection is ranked according to relevance.

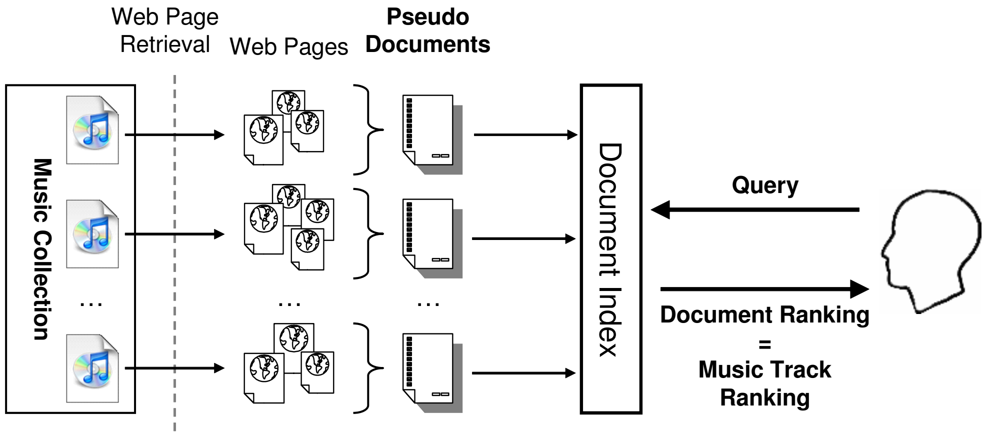


Figure 5.2: Schematic overview of the pseudo document approach to indexing and retrieval.

is directly represented by a single text document within the index. A schematic overview of the pseudo document approach can be seen in Figure 5.2.

For retrieval, relevance of music pieces with respect to a query q is obtained by querying the pseudo document index with q. Relevance of the indexed pseudo documents with respect to q is obtained by calculating the Lucene scoring function $score(q, \psi_m)$ for each document ψ_m that contains at least one term from the query q (cf. Equation 3.4). As a result, a ranking of pseudo documents is obtained, i.e., a sequence of $\psi_i, i \in M$. This ordered list of ψ_i is directly interpretable as a ranking of i where $i \in M$, i.e., a ranking of music pieces since there exists a direct and unique mapping between music piece i and text document ψ_i.

The advantage of this approach is that the problem of music indexing is transferred to the text domain. Therefore only a standard text indexing technique is needed which is favourable since retrieval of text documents is a well-researched topic and retrieval systems are optimised in terms of performance.

5.3.2 Web-Document-Centred Approach

This alternative method to obtain a relevance ranking of music pieces with respect to a given text query operates directly on the index of individual Web pages I. Instead of just joining all available information for one music piece without differentiating and constructing pseudo documents that are heterogeneous in structure and content, a simple ranking function is introduced to propagate relevance information from a Web document ranking to scores for individual music pieces.

More precisely, relevance of a music piece m with respect to a given query q is assessed by querying I with q (and obtaining a ranking of Web pages according to the Nutch/Lucene scoring function, see Equation 3.4) and applying a technique called *rank-based relevance scoring (RRS)* to the n most relevant Web documents in I with respect to q. RRS exploits the associations between pages and tracks established in the data acquisition step. A schematic overview of this approach can be seen in Figure 5.3.

The idea of the RRS scoring function is that if a Web page is highly relevant to

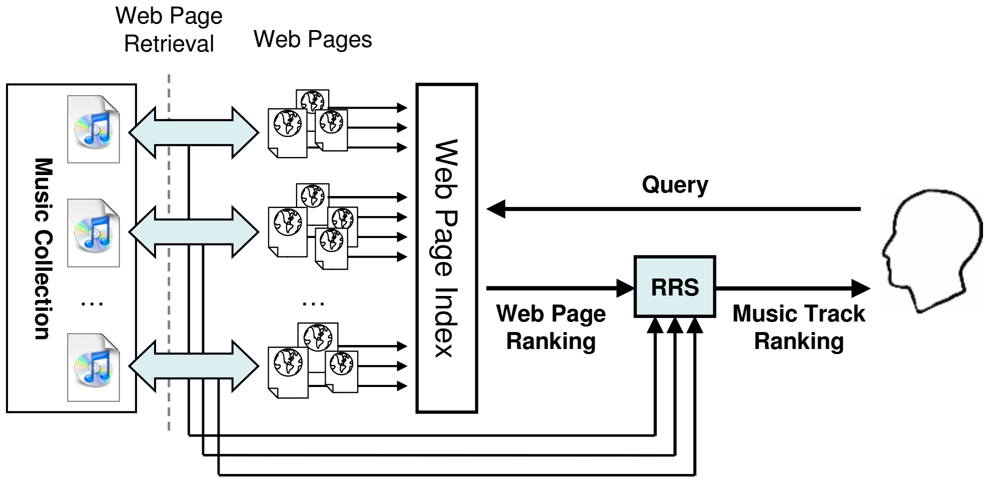


Figure 5.3: Schematic overview of the Web-document-centred approach to indexing and retrieval.

query q, this might also be an indicator that the music piece(s) for which this Web page was relevant in the data acquisition step is also highly relevant. A Web pages that is not as relevant to q is seen as indicator that associated music pieces are also not as relevant to q. This relation is expressed in the RRS scheme by exploiting the rank of a Web page p within the Web page ranking obtained for query q. The relevance score of a music piece is assessed by summing up the negative ranks of all associated Web pages occurring in the page ranking.

$$RRS_n(m, q) = \sum_{p \in D_m \cap D_{q,n}} RF(p, D_{q,n}) \tag{5.1}$$

$$RF(p, D_{q,n}) = 1 + |D_{q,n}| - rnk(p, D_{q,n}) \tag{5.2}$$

Here, n denotes the maximum number of top-ranked documents when querying I, $D_{q,n}$ the ordered set (i.e., the ranking) of the n most relevant Web documents in I with respect to query q, and $rnk(p, D_{q,n})$ the rank of document p in $D_{q,n}$. For music retrieval, the final ranking is obtained by sorting the music pieces according to their RRS value.

Note that compared to the first RRS formulation in [Knees et al., 2008a], the additional parameter n is introduced to limit the number of top-ranked documents when querying the page index. For large collections, this is necessary to keep response times of the system acceptable.[3]

[3]Furthermore, as suggested by [Turnbull et al., 2008a, Barrington et al., 2009], a weight-based version of relevance scoring (WRS) that incorporates the scores of the Web page retrieval step rather than the ranks, has been explored. In preliminary experiments this modification performed worse and is therefore not further considered. Possible explanations for these inconsistent results are the differences in the underlying page scoring function and the different sources of Web pages (cf. [Turnbull et al., 2008a]).

5.4 Selection of Pages Using Filtering Techniques

In the data acquisition step, additional constraints have been applied to the query strings to limit the set of retrieved data to texts that contain useful terms for indexing the corresponding music pieces. Although these constraints are useful in acquiring more targeted Web pages, still not all retrieved information is relevant. Incorporating irrelevant information, however, likely affects the indexing quality. The goal of this section is to identify and filter out texts that lead to retrieval of irrelevant tracks.

As can be seen from evaluations previously published in [Knees et al., 2008a] and [Knees et al., 2009], precision, i.e., the fraction of relevant results among all retrieved documents (cf. Section 5.6.2), hardly ever exceeds 30% using the above mentioned scoring approaches on the acquired Web data. That is, rankings usually contain more than twice as many irrelevant pieces as relevant ones. Based on this, subsequent steps such as combination with audio similarity (cf. Section 5.5) may also suffer from erroneous input. Clearly, the underlying Web pages are responsible for the high number of irrelevant pieces. For indexing as described above, all pages returned by the Web search engine are considered relevant, irrespective of whether they actually contain information about or descriptions of the corresponding music piece or artist. Furthermore, the page indexer does not distinguish between text that occurs in the "main part" of the Web page and text that is used for navigation or links to stories about other, completely unrelated artists. Thus, to improve precision of the retrieved set of music pieces, in the following, four different filtering approaches to remove noisy information and documents are proposed.

5.4.1 Unsupervised Filtering

The characteristic of these filters is that they aim at identifying misleading texts without information from external sources. Hence, they can be applied to the index I directly after building it. The first filter does not remove full documents from the index, but tries to identify those portions within the indexed text that do not contain specific information. The second approach identifies and removes complete documents. Both strategies can be applied prior to the *pseudo document vector space model approach* (Section 5.3.1), as well as the *RRS-based document-centred approach* (Section 5.3.2).

5.4.1.1 Alignment-Based Noise Removal

As mentioned earlier, most indexed Web pages contain not only relevant and interesting information (if any at all). Almost every page contains a site-specific header, navigation bar, links to related pages, and copyright disclaimers, frequently automatically generated by a content management system (cf. [Yi et al., 2003, Debnath et al., 2005]). Especially on music pages, these segments often feature lists of other music artists, genres, or tag clouds to facilitate browsing. This surrounding information is usually not relevant to the associated music piece and should thus be ignored.

Removal of this kind of text is the aim of this filter which is called *alignment-based noise removal (ANR)*. Since large parts of the surrounding text remain the same for most pages within a Web domain, we can identify redundant segments by comparing several texts from the same domain. Coherent parts are most likely to be non-specific for a given music piece and can therefore be removed. To this end, the redundant parts of Web sites are identified by applying *multiple sequence alignment* as described in Section 3.1.5. This technique was originally used to extract lyrics from multiple Web sources by matching coherent parts and preserving overlapping segments (cf. [Knees et al., 2005]). Here, it is applied for the converse purpose, namely to identify and remove redundant segments.

To apply the filter, all documents belonging to the same domain are collected. Especially for blogs, the domain alone does not indicate similarly structured pages — different blogs are typically accessible via separate sub-domains (as is the case, e.g., for *blogspot.com*) — the sub-domain is kept if the host section of the URL contains the word "blog". For domains that occur only up to five times in the page index, no filtering is performed. For all other domains, up to eight documents are chosen randomly and used for alignment. From the alignment, all aligned tokens occurring in at least 60% of the aligned texts are chosen. Finally, all text sequences consisting of at least 2 tokens are selected and removed in all Web pages originating from the domain under consideration.

From experiments it becomes clear that this method excludes almost exclusively noise from the data. Frequent patterns over all domains comprise the name of the Website, copyright disclaimers, and login forms (e.g., "Sign In", "Sign Up Now", and co-occurring terms). Structures for navigation and listings for browsing can also be observed frequently.

5.4.1.2 Too-Many-Artists Filtering

With this filter, the goal is to detect indexed pages that do not deal with only one type of music, i.e., pages that provide an ambiguous content and are therefore a potential source of error. Some of these pages can be identified easily, since they contain references to many artists. Hence, the page index is queried with every artist name occurring in the music collection and the occurrences of each page in all result sets are counted. Constructing the filter simply consists in selecting a threshold for the maximum number of allowed artists per page. By experimenting with this threshold, promising results were obtained when removing all pages containing more than 15 distinct artists (cf. tables A.7 to A.12). Throughout the remainder of this thesis, *too-many-artists filtering (2MA)* refers to the removal of pages containing more than 15 artists.

5.4.2 Supervised Filtering

Automatic optimisation of the (unsupervised) Web-based indexing approach is in general difficult since for arbitrary queries there is no learning target known in advance (in contrast, for instance, to the approaches presented in [Turnbull et al., 2007a, Barrington et al., 2009], where the set of possible queries is limited). However, for the task of identifying sources of noise, automatic optimisation approaches are somewhat more promising — provided that a set of potential queries with corresponding relevance judgements is available (for example, ground truth annotations typically used for benchmarking, cf. the orange box in the schematic overview in Figure 5.1). In the following, such annotations are available through the two test collections used for evaluation. Both collections are introduced in more detail in Section 5.6.1. To exploit the available annotations for supervised learning while still being able to evaluate performance with the same set, *cross validation* has to be carried out.

In general, to obtain a potential set of test queries Q and — associated to that — judgements on the relevance of each piece in a music collection with respect to the queries, one may either fall back on manual annotations (if present; as is the case for the CAL500 set, cf. Section 5.6.1.2), or use community-based labels (as in the case of the c35k test collection, cf. Section 5.6.1.1). Such annotations, or tags, can be used directly as test queries to the system and serve also as relevance indicator (i.e., a track is considered to be relevant for query q if it has been tagged with tag q). In the supervised approaches, these annotations are used to allow for automatic selection of the Web data used to index the music collection.

The idea of the supervised filtering is that by observing performance on a given set of queries, it should be possible to identify Web pages responsible for introducing errors and to exclude them from future retrieval tasks (as is done in the first supervised filter presented in Section 5.4.2.1). Having these negative examples, one may even train an automatic classifier from these instances to learn to identify misleading Web pages. This should allow to also exclude other erroneous data (as is done in the second supervised filter presented in Section 5.4.2.2). Ultimately, for both filters, the intention is to obtain also better results on previously unseen queries. This is based on the assumption that documents responsible for introducing noise to a music piece ranking contain erroneous (at least ambiguous) information and are likely to introduce noise to other queries too.

The two supervised page filtering approaches presented both build upon the RRS scoring function presented in Section 5.3.2. In theory it would be possible to also exploit these filtering steps for usage in the pseudo document approach. To this end, a hybrid approach would have to be followed, for instance by learning to filter pages based on RRS scores and applying the resulting filters prior to constructing the pseudo documents. However, in the following only the applicability of the presented filters to the RRS-based approach is investigated to keep the methods conceptually separated.

5.4.2.1 Query-Based Page Blacklisting

Following the general idea outlined in Section 5.4.2, a simple filter that blacklists Web pages contributing more negatively than positively to query results is constructed. Note that blacklisting in this context is synonymous to removal from the Web page index. By monitoring the performance (the influence, rather) of indexed Web pages over the course of different test queries, a list of those Web pages that were responsible for introducing irrelevant tracks into the rankings is created. For future queries, these Web pages will be excluded. As a consequence of this method, only observed examples can be judged, i.e., no generalisation is made to predict the influence of Web pages not involved through one of the test queries.

To rate a page p, a simple score based on RRS is calculated:

$$S_n(p) = \sum_{q \in Q} \left(\sum_{m \in M_p \cap T_q} RRS_n(m,q) - \sum_{m \in M_p \cap \overline{T_q}} RRS_n(m,q) \right) \tag{5.3}$$

where Q denotes the set of all available queries/annotations, M_p the set of all music pieces associated with page p, T_q the set of all pieces annotated with q (i.e., relevant to query q), and $\overline{T_q}$ its complement (i.e., all music pieces not relevant to q). Informally speaking, over all queries, the sum of RRS scores contributed to negative examples is subtracted from the sum of RRS scores contributed to positive examples.[4] Finally, all Web documents p with $S_n(p) < 0$ are removed, i.e., all documents that contributed more negatively than positively over the course of all queries.

5.4.2.2 Query-Trained Page Classification

While the *query-based page blacklisting* filter represents (if any) just the "laziest" form of machine learning (i.e., merely recognising instances without any kind of generalisation), this filter aims at learning to automatically classify Web pages as either "positive" (keep) or "negative" (remove). Hence, it should be better suited to deal with new queries that provoke previously unseen (and thus unrated) Web pages.

Similar to the query-based page blacklisting filter, the $S_n(p)$ value as defined in Equation 5.3 is used for determination of positive and negative training instances for the classifier. More precisely, whereas in the query-based page blacklisting filter in Section 5.4.2.1 this score only served to identify and remove negative instances, here, the $S_n(p)$ score is used to determine whether a seen Web page can serve as a "positive" or as a "negative" training instance. Naturally, pages p with $S_n(p) > 0$ serve as positive examples and, correspondingly, pages with $S_n(p) \leq 0$ as negative examples. Additionally, only pages that appear in the result sets of at least two queries are considered. Furthermore, since the number of training instances (i.e., the number of evaluated Web pages) gets very high for larger values of n (i.e., the number of top-ranked Web pages incorporated into RRS), which slows down training of the classifiers applied, and since there are usually significantly more negative than positive examples available, the number of total training examples is limited to 2,000. To this end, a random sampling is performed on the positively, as well as on the negatively rated Web page sets to construct a balanced training set with equally distributed classes. In case not enough positive instances are available, further negative examples are added to the training set and a cost-sensitive meta-classifier is applied to raise importance of positive instances (misclassification of positive instances is penalised by the ratio of negative to positive examples). For the remaining examples (i.e., negative as well as positive examples that are not used as training instances), a black- and whitelisting approach is applied, i.e., pages with a negative score are just removed and pages with a positive score will be accepted without performing additional classification on them.[5]

[4] Note that this formulation exhibits some similarities with the relevance feedback formula by [Rocchio, 1971] (cf. Section 6.3). Indeed, the idea of rating pages based on performance judgements from other queries can be considered a relevance feedback process.

[5] The first version of the query-trained page classification presented in [Knees et al., 2010] de-

As feature representation for Web pages, characteristic values such as the length of the page's (unparsed) HTML content, the length of the parsed content, the number of different terms occurring on the page, the number of associated music pieces (i.e., $|M_p|$), the number of contained artist names (cf. Section 5.4.1.2), as well as ratios between these numbers are chosen. Furthermore, title and URL of the pages are utilised to serve as very short textual representations of the respective page (with a smaller vocabulary), converted into a term vector space and added as numerical features. For classification, inside the cost-sensitive wrapper, the *Random Forest Classifier* from the WEKA package is applied with 10 trees (cf. Section 3.1.4). The result of this step is an automatic classifier, that predicts whether an indexed Web page should be kept in the index (i.e., the Web page is likely to contribute mostly relevant tracks to rankings) or removed (likely to contribute mostly irrelevant tracks).

5.5 Post-Hoc Re-Ranking Based on Audio Similarity

Since the methods presented so far are solely based on texts from the Web, important acoustic properties may be neglected and the indexing may suffer from effects such as a popularity bias. Foremost, tracks not present on the Web are excluded by this approach. By incorporating audio similarity into the retrieval process, the methods proposed in this section aim at remedying these shortcomings and improving ranking quality. A combination with audio similarity information should especially enforce the inclusion of lesser known tracks (i.e., tracks from the "long-tail") into the search results.

According to [Snoek et al., 2005,Barrington et al., 2009], the presented approach can be considered a *late fusion* approach, for it modifies the ranking results obtained from the Web-based retrieval. Alternatively, so-called *early fusion* approaches are conceivable, e.g., by incorporating the audio similarity information directly into the RRS weighting scheme like proposed in [Knees et al., 2009]. However, since the proposed re-ranking approach is indifferent about specifics of the underlying ranking algorithm (and therefore can be applied to both, pseudo document and RRS-based retrieval), only this type of audio-based ranking modification is discussed.

In the following, a re-ranking approach called *post-hoc audio-based re-ranking (PAR)* is described that uses audio-based similarity information in an unsupervised manner. Basically, the algorithm incorporates the idea of including tracks that sound similar to tracks already present in a given relevance ranking R of music pieces. Following the path outlined in Figure 5.1, it is assumed that R (which is an ordered subset of M) has been obtained via one of the two text-based retrieval methods presented in Section 5.3. Starting from such a text-based ranking and having access to audio similarity information for the tracks in R, the new (re-ranking) score of any track m is calculated by summing up a score for being present in the text-based

fines positive and negative examples by considering only pages that have either contributed exclusively positively or exclusively negatively, respectively, i.e., positive examples are defined as $\{p \,|\, p \in D_{q,n}, \forall q \in Q : M_p \cap \overline{T_q} = \emptyset\}$ and negative as $\{p \,|\, p \in D_{q,n}, \forall q \in Q : M_p \cap T_q = \emptyset\}$. This may result in very unbalanced training sets and — for high values of n — also in bad classification results. Obviously, the cost-sensitive meta-classifier applied can not compensate for this sufficiently. For this reason, the definition given here enforces a more balanced set on the expense of introducing less clearly defined page ratings.

ranking R and scores for being present within the k nearest audio neighbours of tracks in the text-based ranking R. For audio similarity computation, the measure that combines MFCC-based similarity with Fluctuation-Pattern-based similarity is applied (see Section 3.2.4). Note that for constructing the ranking of acoustically nearest neighbours for a song, all other songs by the same artist are excluded (a step known as *artist filtering*, cf. [Flexer, 2007]), since this similarity is already represented within the Web features.

The post-hoc audio-based re-ranking scores are calculated as:

$$PAR(m, R) = \sum_{l \in (m \cup A_m) \cap R} RF(l, R) \cdot NF(m, l) \tag{5.4}$$

$$NF(m, l) = \alpha \cdot c(m, \{l\}) + W_{gauss}(rnk(m, N_{l,k})) \cdot c(m, N_{l,k}) \tag{5.5}$$

$$W_{gauss}(i) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(i/2)^2}{2}} \tag{5.6}$$

where $RF(l, R)$ is calculated according to Equation 5.2 [6], $N_{l,k}$ denotes the k nearest audio neighbours of track l (i.e., the k most similar tracks to l, sorted in ascending order according to $dist_{comb}$, cf. Equation 3.10), and A_m the set of all tracks l that contain m in their nearest audio neighbour set, i.e., all l for which $m \in N_{l,k}$. $c(x, N)$ is a function that returns 1 iff $x \in N$ and 0 otherwise. The function W_{gauss} represents a Gaussian weighting of the audio neighbours depending on their rank that was introduced because it yielded best results when exploring possible weightings. Parameter α can be used to control the scoring of tracks already present in R. Note that for $k = 0$, R remains unchanged.

5.6 Evaluation

The proposed algorithms can be used to obtain rankings as shown exemplary in Table 5.1. Since assessing the relevance of each track in the returned rankings manually is infeasible for large collections and a substantial amount of queries, for evaluation, automatic methods that compare results against a “ground truth” have to be applied.

This section deals with automatically evaluating the proposed retrieval methods as well as assessing the effects of the filtering and re-ranking extensions on a larger scale. To this end, the different methods are applied to two test collections with different characteristics. First, these collections are presented (Section 5.6.1). Second, the evaluation measures used to estimate the performance of the methods are reviewed (Section 5.6.2). Sections 5.6.3 through 5.6.5 deal with the evaluation of the methods and extensions. Section 5.6.6 investigates the effects of audio-based re-ranking on tracks from the “long-tail”.

[6] Note that RF as described in Equation 5.2 takes a Web page and a ranking of pages as parameters. In this context, it is used for calculating a score based on a track’s position within a ranking of tracks. Apart from the different nature of parameters, the calculation is done in the same way.

5.6.1 Test Collections

The first collection is a large real-world collection and contains mostly popular pieces. The second collection is the CAL500 set, a manually annotated corpus of 500 tracks by 500 distinct artists [Turnbull et al., 2008b]. After discussing preprocessing steps applied to the collections prior to the Web data acquisition step, both test collections are described in more detail.

To obtain descriptors for the tracks in an mp3 collection, the values of the fields "artist", "album", and "title" are extracted from the ID3 tags (cf. Section 5.2). Very commonly, additional information is included in these tags, e.g., tracks that are a collaboration of two artists often contain the second artist in the title (indicated by *feat.*, *and*, *with*, etc.) or both artists are mentioned in the artist field. Other meta-information (e.g., to indicate live versions or remixes) can also be found. To avoid too constrained queries in the next step, this extra information is removed. One drawback of this preprocessing step is that it affects artists like "Ike & Tina Turner", who are afterwards represented only by "Ike".

Based on the meta-tags, pieces that are likely to contain only speech are also ignored (e.g., in Rap music this is often indicated by the word *Skit*) as well as all tracks named *Intro* or *Outro* and tracks with a duration below 1 minute. Furthermore, all duplicates of tracks are excluded from the next steps to avoid unnecessary similarity computations and redundancies (different versions of tracks could be displayed, for example, as alternatives in the retrieval results). Among all duplicates, the version containing no meta-information like *live* or *remix* is chosen for further processing.

5.6.1.1 c35k Collection

The *c35k* collection is a large real-world collection, originating from a subset of a digital music retailer's catalogue. The full evaluation collection contains about 60,000 tracks. Filtering of duplicates (including remixes, live versions, etc.) reduces the number of tracks to about 48,000. As groundtruth for this collection, Last.fm tags are utilised. Tags can be used directly as test queries to the system and also serve as relevance indicator (i.e., a track is considered to be relevant for query q if it has been tagged with tag q). Out of the 48,000 tracks, track-specific Last.fm tags are available for about 35,000. However, most of these tags are not suited for evaluation purposes (i.e., as test queries). For instance, many of the track-specific tags consist only of the name of the corresponding artist. Since retrieval using explicit track meta-data is not the objective of this chapter and manual cleaning of tag-sets for 35,000 tracks is rather labour-intense, for selection of an appropriate set of test queries, Last.fm's list of top tags is exploited. Top tags represent the most popular (i.e., most frequently used) tags on Last.fm. The majority of the retrieved 250 top tags consists in genre or style descriptors such as *rock*, *electronic*, or *alternative*. Beside these, decades (*60s*, *80s*), adjectives (*awesome*, *beautiful*), or tags used to express personal experiences (*favorites*, *albums i own*, *seen live*) can be found. For creating a test query set, starting from the list of top tags, all personal tags are removed. Furthermore, redundant tags (such as *hiphop*, *hip hop*, and *hip-hop*) are identified manually and their sets of tagged tracks are harmonised (i.e., any track that is tagged with either of the three variants above is defined relevant for all three). Different morphological forms are retained as distinct queries as long as they translate to different queries after query parsing (cf. 3.1.2, in the example above, *hiphop* translates to a query with one token, *hip hop* to two tokens, and *hip-hop* to a phrase). As result, a set of 223 queries remains. From the 223 tags, further all tags with a number of associated tracks above the 0.95-percentile and below the 0.05-percentile are removed, resulting in 200 test queries. Table A.2 shows the used 200 queries. In addition, for each query, the number of relevant tracks in the collection is given. A common way to increase the number of tagged examples is to use artist-specific tags if no track-specific tags are present [Knees et al., 2007a, Turnbull et al., 2008a]. Since, in the presented indexing approach, tracks by the same artist share a large portion of relevant Websites, this substitution is omitted to avoid overestimation of performance.

5.6.1.2 CAL500 Set

The CAL500 set is a highly valuable collection for music information retrieval tasks [Turnbull et al., 2008b]. It contains 500 songs (each from a different artist; without album information) which were manually annotated by at least three reviewers. Annotations are made with respect to a vocabulary consisting of 174 tags describing musically relevant concepts such as genres, emotions, acoustic qualities, instruments, or usage scenarios. Although the presented indexing approaches are in principle capable of dealing with arbitrary queries and large and varying vocabularies, some of the CAL500 tags are not directly suited as query. Especially negating concepts (e.g., *NOT-Emotion-Angry*) can not be used. Hence, all negating tags are removed. Furthermore, redundant tags (mostly genre descriptors) are joined. For tags consisting of multiple descriptions (e.g., *Emotion-Emotional/Passionate*) every description is used as an independent query. This results in a total set of 139 test queries (see Table A.3).

5.6.2 Evaluation Measures

To measure the quality of the obtained rankings and the impact of the extensions as well as different parameter settings, standard evaluation measures for retrieval systems are calculated, cf. [Baeza-Yates and Ribeiro-Neto, 1999], p.74*ff.* In the following, R denotes a ranking (i.e., the ranking to be evaluated) obtained for query q and T_q denotes the set of all pieces annotated with q (i.e., the ground truth of relevance for query q).

For global assessment of the returned result sets, the well-established measures *precision* and *recall* are calculated. Precision is defined as the fraction of the retrieved music tracks that is relevant to the query, whereas recall is the fraction of the relevant pieces that have been retrieved, i.e.,

$$Prec(R,q) = \frac{|R \cap T_q|}{|R|} \tag{5.7}$$

$$Rec(R,q) = \frac{|R \cap T_q|}{|T_q|} \tag{5.8}$$

Since usually for retrieval systems the retrieved documents (in this case, music pieces) are presented according to their relevance (i.e., in a ranking), a system's ability to rank relevant documents to the top of the result list should be rewarded more. Since precision and recall values do not reflect this characteristic, more specific evaluation measures for rankings have been introduced.

For instance, as a very intuitive measure, *precision @ 10 documents (Prec@10)* represents the number of relevant pieces among the first 10 retrieved. This is important since usually search engines display only the 10 top-most results on the first page, i.e., Prec@10 measures how many relevant music pieces can be expected "at first sight". Other single value summaries of ranking performances used in the following are *R-precision (RPrec)* and *average precision (AvgPrec)*, the latter also known as *average precision at seen relevant documents*. R-precision measures the precision at the r^{th} returned document, where $r = |T_q|$, i.e., it measures precision at that point at which in an optimal ranking, precision would be 1.0. Average precision is a measure that favours systems capable of quickly retrieving relevant documents. It is calculated as the arithmetic mean of precision values at all encountered relevant documents (when processing the ranking sequentially).

Formally, the respective definitions of precision @ 10 documents, R-precision, and average precision are as follows (where R_r denotes the r top-most documents of ranking R):

$$Prec@10(R,q) = Prec_{10}(R,q) = \frac{|R_{10} \cap T_q|}{|R_{10}|} \tag{5.9}$$

$$RPrec(R,q) = Prec_{|T_q|}(R,q) = \frac{\left|R_{|T_q|} \cap T_q\right|}{\left|R_{|T_q|}\right|} \tag{5.10}$$

$$AvgPrec(R,q) = \frac{\sum_{m \in R \cap T_q} Prec_{rnk(m,R)}(R,q)}{|R \cap T_q|} \tag{5.11}$$

To observe precision over the course of a ranking, further *precision at 11 standard recall levels* is calculated that allows for standardised plotting of precision vs. recall curves. When examining the ranking, the occurrence of a relevant piece allows to calculate precision at a specific recall level (that depends on the number of relevant documents encountered so far and the overall number of relevant pieces), in the following denoted as $Prec@Rec(s,R,q)$, where s corresponds to the recall level, i.e.,

$$Prec@Rec(s,R,q) = Prec_r(R,q) \iff s = \frac{|R_r \cap T_q|}{|T_q|} \tag{5.12}$$

Since different queries have different numbers of associated relevant pieces, in general, observed recall levels will be distinct for different queries. To assess the quality of an algorithm for a set of queries, i.e., to allow for averaging over multiple queries, an interpolation to standardised recall levels has to be carried out. To this end, precision $Prec@Rec(s_j,R,q)$ at the 11 standard recall levels $s_j, j \in \{0.0, 0.1, 0.2, ..., 1.0\}$ is interpolated according to

$$Prec@Rec(s_j,R,q) = max_{s_j \leq s \leq s_{j+1}} Prec@Rec(s,R,q) \tag{5.13}$$

Using this standardisation, an average precision vs. recall curve over all queries can be calculated by averaging Prec@Rec values for each standard recall level. For a single value comparison of different algorithms, additionally, the *area under the precision at 11 standard recall levels curve (AUC)* is calculated. Conceptually, this measure corresponds to the average precision value.

In general, for evaluation, no single queries are examined but the average (mean) value over all queries. Furthermore, to compare different settings, statistical significance testing is carried out. Since the resulting values of evaluation measures over all queries are not normally distributed, a non-parametric test is applied to the above mentioned measures. More precisely, the *Friedman test* with attached post-hoc tests at a significance level of $\alpha = 0.01$ is performed to compare multiple settings, cf. [Hollander and Wolfe, 1999]. For evaluation of the supervised extensions (filtering as well as combination), a 10-fold cross validation is performed on the test collections, i.e., in each fold, 90% of the queries are used to train the filters which are then applied and evaluated on the remaining 10%.

5.6.3 Web Search Engine Impact

Since a Web search engine is the source for the underlying Web data, and therefore the primary prerequisite for the proposed approaches, its impact is evaluated first. For comparison, Web data is acquired by using two distinct Web search engines, namely *Google* and the lesser-known French search engine *exalead*[7]. This is important to assess the influence of the Web indexing and retrieval algorithms on downstream approaches such as those presented here as well as for estimating the consequences of a possible exchange of this commercial component with a (music-focused) Web index obtained by a self-developed Web crawler in the future. Figure 5.4 shows precision at 11 standard recall level plots for two different RRS setting ($n = 200$ and $n = 10000$) and the pseudo document approach on both test collections to compare the performance of Google and exalead. The given baselines depict collection-specific references that are obtained by averaging over evaluation results of random permutations of the full collection. Tables 5.2 and 5.3 show additional evaluation measures for the same settings.

As can be seen, Google clearly outperforms exalead in all settings and for both collections. Only for (global) precision, does exalead yield better results, leading to the assumption that the Web pages retrieved via exalead are high in precision but worse in recall when compared to Google. Not surprisingly, this indicates that the quality of the algorithm for retrieving the underlying Web data, as well as the size of the Web index, play central roles for subsequent steps.

Furthermore, the conducted evaluations give other interesting insights. For the c35k collection, it can be seen that for high values of n, i.e., the number of Web pages incorporated into the scoring, the RRS approach clearly produces better rankings than the pseudo document method (with respect to presenting relevant results earlier in the ranking).[8] This is further confirmed by the values of precision @ 10 documents, R-precision, and average precision as displayed in Table 5.2. However,

[7] `http://www.exalead.fr`

[8] Note that the recall value of pseudo document retrieval represents the upper bound of recall for all RRS-based approaches.

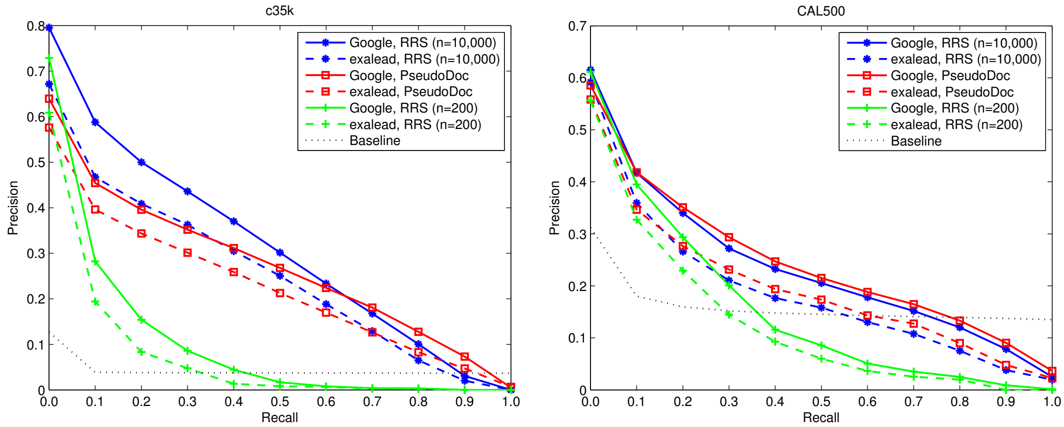


Figure 5.4: Precision at 11 standard recall level plots to compare the impacts of Google and exalead on the c35k collection (left) and the CAL500 set (right).

	$\mathbf{RRS}_{n=200}$		$\mathbf{RRS}_{n=10000}$		**PseudoDoc**	
	Google	exalead	Google	exalead	Google	exalead
Rec	**18.67**	13.40	**80.50**	72.48	**93.66**	87.20
Prec	**23.77**	**23.01**	7.29	**8.81**	4.27	**5.52**
Prec@10	**47.75**	33.80	**57.45**	42.55	**39.25**	34.95
rPrec	**14.22**	10.76	**35.20**	30.25	**30.78**	26.72
AvgPrec	**8.23**	5.34	**29.98**	23.97	**25.97**	21.16

Table 5.2: IR measures for comparison of Google vs. exalead on the c35k collection averaged over all test queries. Bold entries indicate the significantly better result between Google and exalead (or, in case of both entries being printed in bold, no significant difference).

	$\mathbf{RRS}_{n=200}$		$\mathbf{RRS}_{n=10000}$		**PseudoDoc**	
	Google	exalead	Google	exalead	Google	exalead
Rec	**38.63**	33.42	**73.27**	61.43	**81.15**	70.52
Prec	**19.15**	**18.72**	14.56	**15.45**	14.50	**15.08**
Prec@10	**30.60**	**28.63**	**33.62**	**30.43**	**30.72**	**26.47**
rPrec	**21.58**	18.77	**25.06**	21.12	**25.77**	21.31
AvgPrec	**13.84**	10.85	**21.77**	16.96	**22.66**	17.57

Table 5.3: IR measures for comparison of Google vs. exalead on the CAL500 set, cf. Table 5.2.

for smaller values of n, performance of RRS is worse than of the pseudo document ranking. In this context it should be mentioned that large numbers of n are factually only of theoretical interest, since RRS calculations have to be performed in addition to page retrieval at query time. Therefore, RRS performs not as efficiently as traditional retrieval approaches. Furthermore, the fact that retrieval systems are not optimised to quickly find the, e.g., $10,000^{th}$ most relevant result for a query contributes to that disadvantage. To assess the influence of the parameter n on the RRS approach in more detail, performance is evaluated systematically for $n \in \{10, 20, 50, 100, 200, 500, 1000, 10000\}$. These detailed results can be found in tables A.5 and A.6 for the c35k collection and the CAL500 set, respectively.

For the CAL500 set, the results exhibit a slightly different picture. First, results retrieved at the top of the rankings (especially the top 10) are of about the same quality for all settings, i.e., $\text{RRS}_{n=200}$ performs similar to $\text{RRS}_{n=10000}$ and pseudo document retrieval. As the ranking progresses, quality of $\text{RRS}_{n=200}$ drops quickly due to the low recall caused by the small value of n. In contrast to the c35k collection, for CAL500, the pseudo document approach performs equally well or even slightly better than $\text{RRS}_{n=10000}$ (cf. R-precision and average precision in Table 5.3).

5.6.4 Page Filtering Impact

To evaluate the impact of the proposed page selection techniques on the ranking quality, again systematic experiments have been conducted. Detailed results for a variety of settings can be seen in tables A.13 to A.16. Figure 5.5 shows evaluation results on both evaluation collections for $\text{RRS}_{n=500}$. The general trends are very consistent for both search engines used. However, different observations can be made for the two different test collections.

In general, on c35k, all filters and combinations thereof yield better ranking results than the unfiltered RRS approach. Furthermore, both supervised filtering methods are clearly superior to the unsupervised filters. A combination of those, however, decreases performance (except for overall precision, e.g., for a combination of unsupervised filters with query-based page blacklisting (QB2), cf. Table A.13 and Table A.14). For the alignment-based noise removal (ANR), slight improvements, especially for precision, r-precision, and average precision can be observed. However, in the Friedman test these results are not significant. For recall and precision @ 10 documents a significant drop in performance when using Google but also a significant increase when using exalead becomes apparent.

The too-many-artists filter (2MA) outperforms the unfiltered RRS significantly in terms of precision and average precision for smaller values of n using Google. For exalead, where precision is initially already higher, this finding can not be confirmed. Not surprisingly, a decrease is most clearly visible for recall. In addition, the combination of both unsupervised filters is evaluated (A2). Results are rather inconsistent, in the plots in Figure 5.5, the combination yields overall better or equal results. For the averaged single value summaries in tables A.13 and A.14, the general observation is that a high degree of uninformed filtering can be too much and may affect results.

With the exception of recall for query-trained page blacklisting (QB), both *supervised approaches*, i.e., QB and query-based page classification (QC), are constantly in the best performing group or at least significantly better than the unfil-

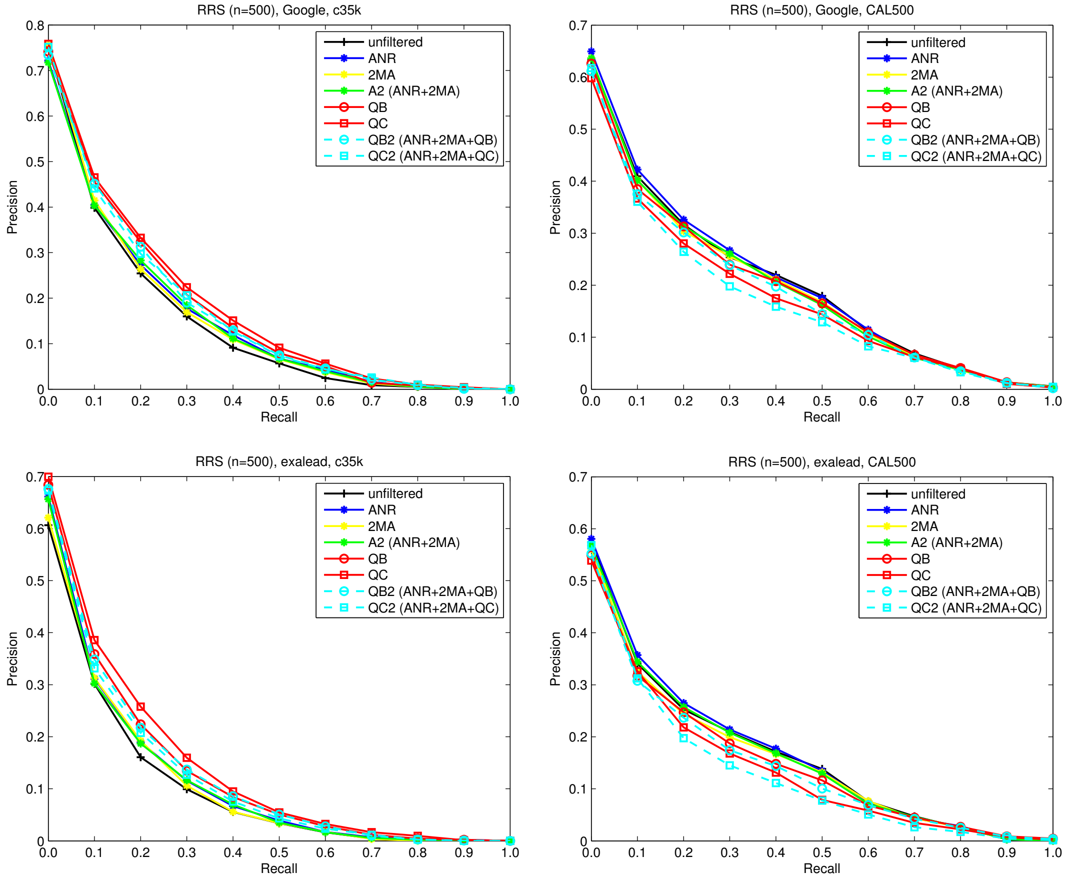


Figure 5.5: Impact of filters and filter combinations on the $\text{RRS}_{n=500}$ setting using Google (top row) and exalead (bottom row) on the c35k collection (left column) and the CAL500 set (right column).

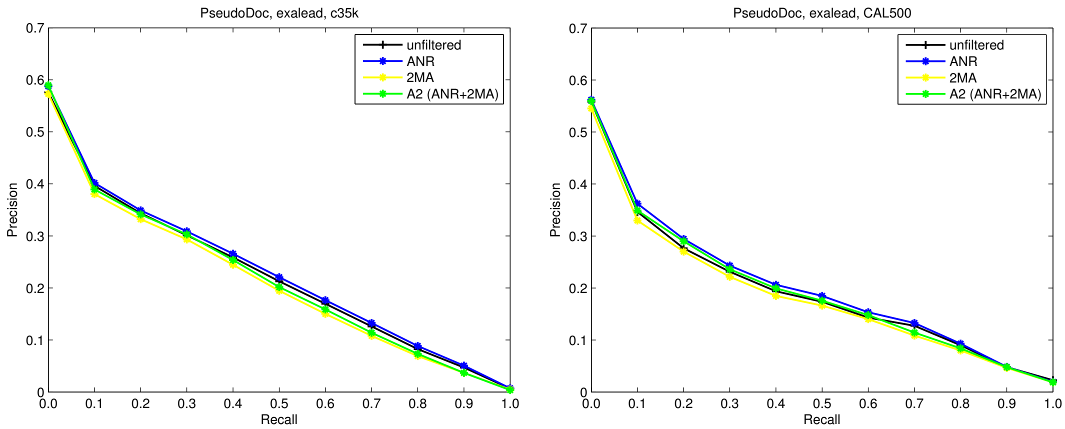


Figure 5.6: Impact of unsupervised filters on pseudo document retrieval using exalead on c35k (left) and CAL500 (right).

tered method. For specific settings, some results report on definitive improvements, for instance, precision @ 10 seen documents for $n = 10000$ increases by about 5 percentage points, cf. Table A.13. When combining supervised with unsupervised approaches (QB2, QC2), one is confronted with ambiguous results. Certainly, the tendency of getting more precise results (going along with a loss in recall) is also present here. In contrast to the combination of both unsupervised filters, averaged single value summaries suggest a potential increase in performance, whereas the precision at 11 standard recall level plots suggest an overall decrease for combination of supervised with unsupervised filters.

For the CAL500 set, results are very different and rather disappointing. No proposed filter can significantly improve results (except for precision of the supervised filters with high values of n, which go along with a dramatic loss in recall due to a very high number of excluded pages, cf. Table A.15 and Table A.16). Only ANR seems to have a positive impact on the ranking quality (cf. Figure 5.5). The reasons are not really clear. One possibility could be that in the case of the c35k set with associated Last.fm tags, the approaches benefit from the inherent redundancies in the tags/queries (e.g., *metal* vs. *black metal* vs. *death metal*). In the case of the CAL500 set, queries exhibit no redundancy, as the set is constructed to describe different dimensions of music. However, this would only affect the supervised filters.

Another explanation could be that the CAL500 page indices contain considerably fewer pages than the c35k indices (for Google, e.g., approx. 80,000 pages for CAL500 vs. approx. 2 million pages for c35k). First, and also in the light of the fact that the CAL500 set has been carefully configured (i.e., especially the fact that only one track per artist is contained, which leads to more track specific pages on the artist level, compared to the other tracks) it seems possible that the index does not contain so many noisy pages. Hence, the proposed strategies do not work here. Second, since the index is rather small, removal of a relatively high number of pages has a higher impact on the overall performance. This becomes especially apparent when examining the results of the supervised approaches for high n. Apart from the results, it should be noted that the CAL500 set is without doubt very valuable for research (high quality annotations, freely available, etc.) but at the same time, it is a highly artificial corpus which can not be considered a "real-world" collection. Hence, some "real-world" problems maybe can not be simulated with such a small set.

For pseudo document ranking, observations over both collections are more consistent. For both, c35k and CAL500, only ANR filtering yields slightly (though not significantly) better results. Further filtering (2MA, A2) can only improve precision at the price of high loss in recall and ranking quality, cf. Figure 5.6. Hence, the application of ANR seems to be a sensible choice before constructing pseudo documents, whereas the 2MA filter seems to affect retrieval performance.

5.6.5 Audio-Based Re-Ranking Impact

The next extension to be evaluated is the audio-based re-ranking component. To this end, re-ranking is applied to RRS and pseudo document indexing in their plain, unfiltered version as well as in their best performing pre-filtered version, i.e., RRS with query-based page classification filtering (QC) and pseudo document indexing with alignment-based noise removal filtering (ANR) enabled. Detailed results for different parameter settings for PAR re-ranking can be found in tables A.17 to A.32.

Besides the trivial findings of increasing recall and decreasing precision, a consistent trend for both search engines and both test collections is that an increasing value of k, i.e., the number of included audio neighbours, leads to increased r-precision and average precision, while precision @ 10 documents remains mostly unaffected (except for large n and PseudoDoc, where results are shattering). For the parameter α it can be observed that higher values — that correspond to a small influence of the audio neighbours — yield better results in terms of ranking quality. In general, RRS ranking benefits from combination with PAR, again with the exception of RRS settings with a high number of pages taken into consideration (large n). As with pseudo document ranking, for large n, re-ranking produces significantly worse results. The finding that an already substantial text-based ranking with high recall usually yields better results than its re-ranked counterpart has already been made in [Knees et al., 2009]. The reason is that audio similarity introduces a lot of noise into the ranking. Hence, to preserve the good performance at the top of the rankings, α should be set to a high value. On the other hand, this prevents theoretically possible improvements.

Comparisons of unmodified ranking approaches, audio-based re-rankings (using the parameter settings $k = 50$ and $\alpha \in 10, 100$), also in conbination with the best performing filtering techniques, i.e., QC for RRS and ANR for PseudoDoc, can be seen as precision-recall plots in figures 5.7 and 5.8. Additional results can be found in tables A.33 to A.40.

The plots suggest that PAR performs very well on all RRS settings. In combination with QC filtering, it benefits from the increased initial ranking quality. However, improvements become mostly apparent for recall levels above 0.1. If α is set to a high value, the top of the ranking remains mostly unaffected by the audio neighbours, i.e., not much difference can be made out between original and re-ranked results. For small values of α, also the top of the ranking may be reordered, leading to loss in precision. For recall levels above 0.1, the overall higher recall, i.e., the fact that in general more and therefore also more relevant pieces are retrieved, has a clearly positive impact on the ranking curve. On the CAL500 set, PAR re-ranking is even capable of compensating for the negative impact of QC filtering.

Applying re-ranking to PseudoDoc rankings on the c35k collection leads to clearly worse results (cf. Figure 5.8). For $\alpha = 10$, the effect is even more severe. On the CAL500 set, things look a bit different. Here, results for PseudoDoc can also be improved by applying PAR re-ranking. Hence, apart from the PseudoDoc setting on the c35k collection, overall it can be stated that a combination of page pre-selection with audio-based post-re-ranking can improve the text-only ranking approach.

5.6.6 Impact of Audio-Based Combination on Long-Tail Retrieval

While until this point audio information has been used mainly to improve retrieval quality, this section takes a look at the potential of audio-based re-ranking for compensating missing Web-based information. More precisely, the usefulness of the combination approaches for tracks from the so-called "long-tail", i.e., tracks that are not present on the Web and can therefore not be indexed with text-only approaches, is estimated.

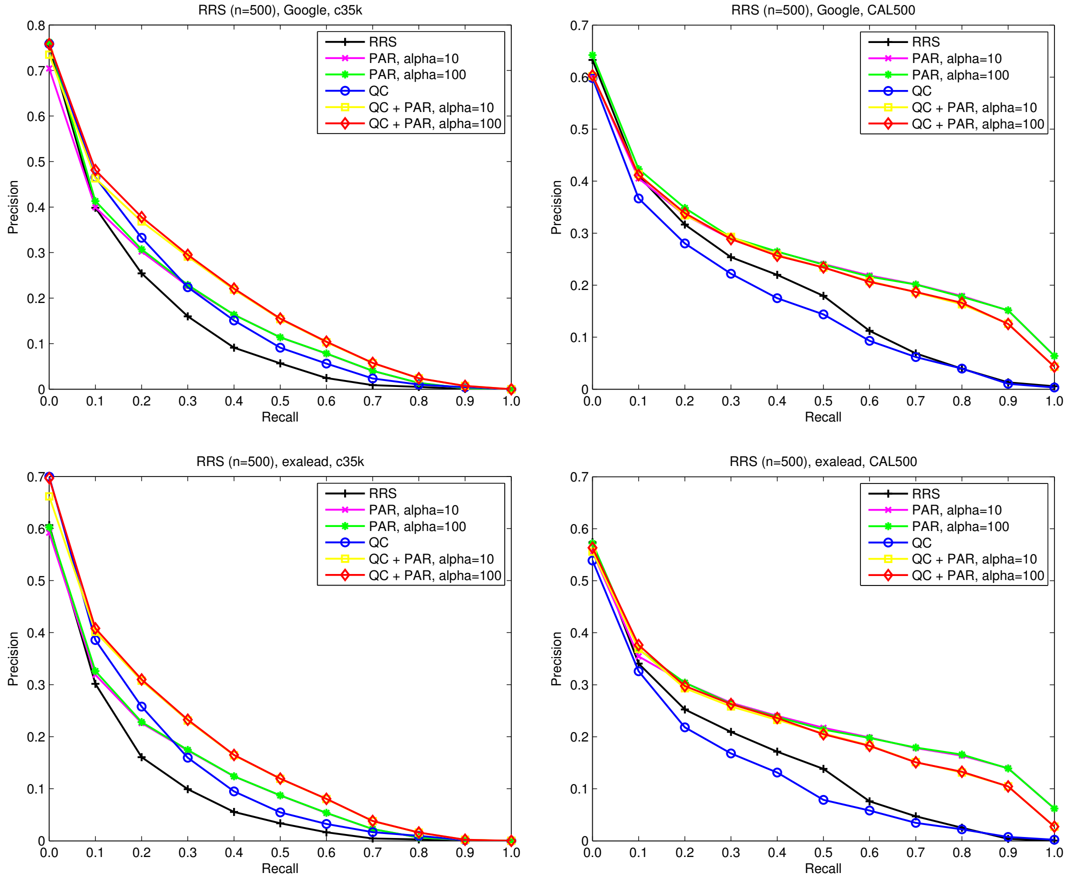


Figure 5.7: Impact of re-ranking on $\text{RRS}_{n=500}$ and QC-filtered RRS using Google (top row) and exalead (bottom row) on the c35k collection (left column) and the CAL500 set (right column).

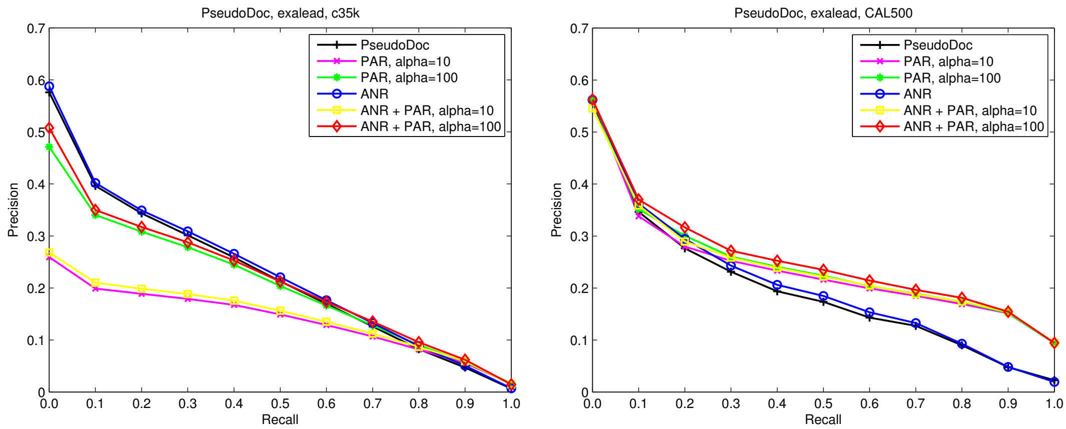


Figure 5.8: Impact of re-ranking on pseudo document retrieval using exalead on c35k (left) and CAL500 (right).

	original	pruned	
	RRS	RRS	PAR
Rec	29.31	14.15	**49.27**
Prec	**20.12**	**19.65**	5.06
AvgPrec	**12.39**	6.30	**9.54**
Overlap	**100.00**	48.24	70.49

Table 5.4: Impact of combination with audio on long-tail retrieval on the c35k collection using Google. The original $\mathrm{RRS}_{n=500}$ setting is compared with the pruned $\mathrm{RRS}_{n=500}$ setting obtained by removing all tracks by randomly selected 50% of all artists and the PAR re-ranked version of the pruned RRS.

	original	pruned	
	RRS	RRS	PAR
Rec	47.36	23.95	**92.02**
Prec	**17.20**	**17.04**	13.43
AvgPrec	14.15	7.71	**18.53**
Overlap	**100.00**	49.36	94.85

Table 5.5: Impact of combination with audio on long-tail retrieval on the c35k collection using exalead, cf. Table 5.4

To assess the potential of audio similarity for this task, a situation where all tracks by 50% of the contained artists in the audio collection have no associated Web data is simulated. The artists whose tracks are excluded from the text-based ranking are chosen at random. In addition to *recall*, *precision*, and *average precision*, an *overlap* with the original, not simulated text-based ranking is calculated to see how well audio similarity can fill the gap of missing Web data. Tables 5.4 and 5.5 show the results of this simulation for the c35k collection (using Google) and the CAL500 set (using exalead), respectively. More detailed results can be found in tables A.41 to A.44

For the RRS approach, dividing the collection in halves leads to halved values for *recall*, *average precision*, and *overlap*, whereas *precision* remains basically unchanged (due to random sampling). It can also be seen that the combination with audio-based similarity can bring back a large portion of relevant tracks. However, it is also apparent that precision suffers significantly. For average precision, PAR re-ranking can also outperform the original ranking, even when building upon the randomly pruned rankings. This conforms with findings made in Section 5.6.5 that PAR is more effective when operating on "sparsely populated" rankings, i.e., rankings with low recall.

5.7 Prototype: The Gedoodle Music Search Engine

For demonstration purposes, a music search engine prototype called *Gedoodle* has been realised via a simple Web front end. The interface design follows the reduced style of the Google Web search front page. Therefore, also the name of the prototype is a reminiscence to Google and also incorporates a musical reference (if maybe not the most positive one): In German, the word Gedudel (pronounced the same way as Gedoodle in English) refers to a bit of annoying, possibly repetitive, piece of music (often in a manner similar to muzak).

Figure 5.9 shows screenshots of the Gedoodle interface in a Web browser and exemplary results pages. For each retrieved track, title, artist name, album title, genre, encoding quality, and length are displayed. Furthermore, a link to the corresponding audio file is provided.

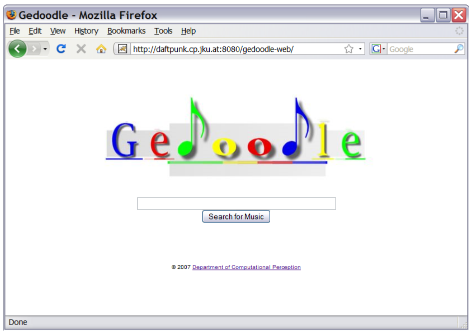
In Figure 5.9(b), results for the query *smooth and relaxing* are shown. As can be seen, tracks by the bands *Count Basic* and *Tosca* are returned as top results. Although not all of the tracks might be considered smooth and relaxing, the overall trend towards *Acid Jazz* music and the *downtempo electronic music* by *Tosca* is correct. Figure 5.9(c) demonstrates a limitation of the current text-based retrieval approaches. For the query *comforting*, one could potentially expect tracks that are similar to those retrieved for the query *smooth and relaxing*. However, since the collection contains the track *Comforting Lie* by the *Ska-Pop* band *No Doubt*, that track is considered most relevant. Furthermore, since the term *comforting* appears also frequently in the context of other tracks from the same album (due to track listings) these tracks are also considered relevant. Currently, such misinterpretations are common errors that can only be dealt with by explicitly modelling occurrences of meta-data in the retrieved Web resources. On the other hand, meta-data search should be provided to the user to allow for retrieval of specific tracks.

The example shown in Figure 5.9(d) demonstrates another quality of "semantic" context-based music search engines not discussed so far, namely the implicit modelling of relations between artists. For the query *damon albarn*, who is most famous for being the lead singer of the band *blur*, without any reference present in the meta-data of the pieces, tracks by *blur*, *Gorillaz*, *Graham Coxon*, and *Fatboy Slim* are returned. All of these results are comprehensible: *Gorillaz* is another project by *Damon Albarn*, *Graham Coxon* was/is a band member of *blur*, and *Fatboy Slim* was a collaborator on a *blur* album.

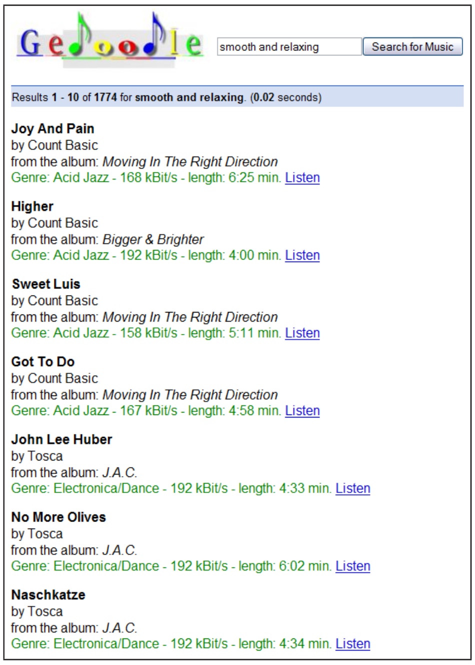
5.8 Recapitulation and Discussion

This chapter has dealt with the indexing and retrieval of music pieces using relevant Web pages obtained via a Web search engine. This allows for building a system that takes free-form text queries for retrieval of music pieces. Two different indexing strategies are presented and evaluated. Furthermore, possible extensions to improve these approaches are investigated, namely techniques for filtering of noisy Web data and a technique to incorporate audio-based similarity information. The latter is also necessary to open up the presented type of text-based retrieval to music pieces from the long-tail, i.e., music pieces without a substantial amount of associated Web data.

For the two different indexing strategies presented, it can be seen that the

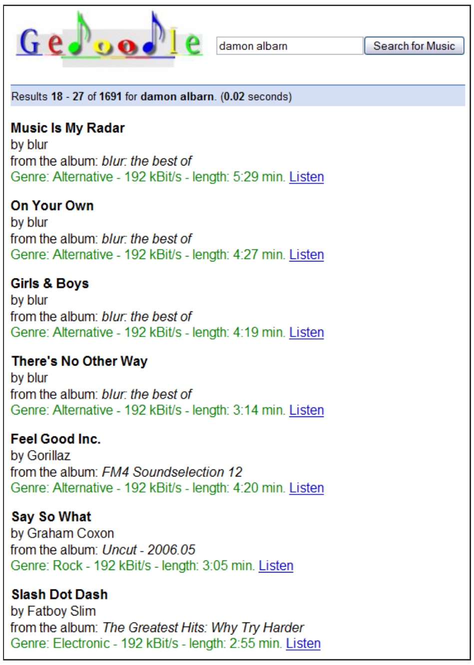


(a) Gedoodle interface



(b) *smooth and relaxing*

(c) *comforting*



(d) *damon albarn*

Figure 5.9: Screenshots of the Gedoodle Music Search Engine. Figure (a) depicts the Gedoodle search interface. Figures (b), (c), and (d) show exemplary results for the queries *smooth and relaxing*, *comforting*, and *damon albarn*, respectively.

document-centred approach that uses the RRS weighting scheme is in principle capable of producing better rankings than the pseudo document indexing approach. However, this superiority becomes only clearly apparent if a very high number of documents is considered for RRS ranking calculation. Since this calculation has to be carried out at query time, in practice, RRS with a high number of considered pages is computationally too expensive. On the other hand, pseudo document retrieval is simpler and more efficient at query time since it translates the music retrieval task to a text retrieval task. Current disadvantages of pseudo document indexing are that the proposed supervised page selection methods for noise removal are not directly applicable and that a combination with audio-based re-ranking may result in a significant decrease of retrieval quality. Thus, the choice which of these indexing approaches should be adopted also depends on the usage scenario. For instance, if retrieval time is not a crucial factor, an RRS-based method will yield better results. In case retrieval time is crucial, the pseudo document method may be better suited. For future developments, to alleviate the problem of long response times for RRS, an incremental version could be designed that modifies the ranking while incorporating more and more Web pages results and stops as soon as the result can be considered stable. For fast retrieval of some first relevant results, this could suffice.

For identification and removal of misleading Web data, two unsupervised and two supervised filtering approaches have been presented. Evaluation shows inconsistent results for two used test collections with different characteristics. From the gained insights it is concluded that the proposed filtering techniques can improve results significantly when applied to large and diverse music collections with millions of Web pages associated. In this case, more or less all of the proposed filtering techniques prove to be useful and improve not only the overall precision but also the ranking of music pieces. By introducing supervised optimisation, there is still more potential to tweak performance. For instance, for the automatic page classification filter, it would not be surprising if a more carefully selected feature set could improve results further. Other classification algorithms could also lead to an improved identification of relevant Web pages and thus improve precision of the overall music piece rankings. In terms of alternative filtering and/or ranking approaches, techniques like *vision-based page segmentation* [Cai et al., 2003] are promising in that they may help in identifying the relevant parts of a Web page. By extracting smaller segments from Web pages, the principle of the RRS weighting could be transferred to individual "blocks" and scoring could be designed more specifically.

With the presented audio-based re-ranking approach, it could be shown that combining Web-based music indexing with audio similarity in principle has the potential to improve retrieval performance. In its current form, on the other hand, it became also apparent that audio-based re-ranking primarily leads to inclusion of many tracks (including many relevant) into the result set without a positive impact on the ordering in the ranking. Thus, while recall can be increased, audio-based re-ranking as presented here, has not shown to be applicable for quickly retrieving more relevant pieces.

A possible reason may be that the applied audio similarity measure is not suited for the given task. To estimate the potential of the currently applied audio similarity measure, the ground truth annotations of the c35k collection are used. For every test query q, all relevant tracks $t \in T_q$ serve as seed song to find similar songs

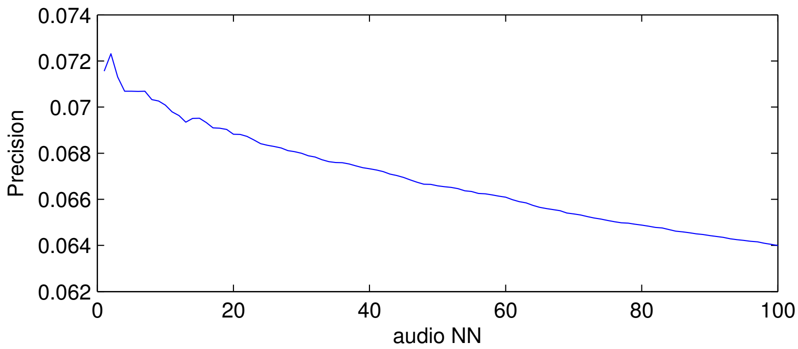


Figure 5.10: Precision at audio-based nearest neighbour for the c35k collection (averaged over all test queries; for every query, average of rankings with each relevant track as seed).

(i.e., as a query-by-example query). On the obtained similarity ranking, precision is calculated at every position $k = 1...100$ with respect to q. Finally, precision at every position is averaged over all t and q. Figure 5.10 shows the result for the c35k collection. Within the top 10 neighbours, a precision of around 7% can be expected in average based solely on the audio similarity. However, it is questionable whether this can be improved as audio similarity measures (statically) focus on specific musical properties, whereas textual queries can be aimed at basically every aspect of music, from different acoustic properties, to cultural context, to completely unrelated things.

In general it has to be stated that proper combination of these two sources is rather difficult since they target different directions and applications. Furthermore, a combination function can not be optimised in advance to suit every potential query, i.e., in contrast to, e.g., [Barrington et al., 2009], automatic learning of proper combination functions (e.g., via machine learning methods) is not applicable for this task since here, no learning target is present. Web-based music indexing as we currently apply it is an unsupervised approach. This is implied by the requirement to deal with a large and arbitrary vocabulary. A possibility to deal with this could be to perform automatic parameter selection for a combination approach as has been shown for the supervised noise filtering approaches.

Chapter 6

Towards Personalised Music Retrieval: Exploiting User Feedback

After presenting techniques for constructing text-augmented browsing interfaces and text-based retrieval systems for music, this chapter discusses the potential of combining the used descriptors with usage information to develop more personalised and user-oriented music services and allow for *adaptive searching*. More precisely, this chapter proposes to incorporate a *relevance feedback* mechanism into the vector space model used in [Knees et al., 2007a] to exploit explicit user feedback. The aim of this extension is to be able to compensate for insufficient text queries (as well as insufficient initial mappings to music tracks) and therefore to better meet the user's information need. Besides automatic evaluation of the proposed approach, a user study is presented which gives further insights into users' search behaviours. Finally, the *Search'n'Select* prototype is presented that utilises query-based retrieval strategies to support the user interactively in a targeted browsing process.

6.1 Motivation

Although the methods presented in the preceding chapters are capable of capturing the cultural context of music pieces to some extent, the presented methods still exhibit many limitations. One obvious problem is that they leave ample space for improvement in terms of retrieval quality. Furthermore, apart from technical shortcomings, it has to be kept in mind that users are actually not accustomed to use free-form text input to search for music. Even if these issues could be sorted out in the near future, the inherent problem of individual concepts and intentions behind the issued queries remains. For example, different users will have different expectations of the pieces that should be returned for the query *Folk*. Some users may aim at retrieving music pieces from American singers and songwriters, while others may intend to find all sorts of folkloristic music. Another example would be the query *Indian music* that could target Sitar music, as well as Bollywood movie soundtracks, as well as traditional native American music. While these user specific interests may not be adequately expressible via a query, getting explicit feedback on the relevance of the retrieved pieces from the users can give extremely valuable information to disambiguate query meaning and clarify the original intention.

To address these limitations, in this chapter, it is proposed to incorporate the relevance feedback method by [Rocchio, 1971] to adapt the retrieval of music pieces to

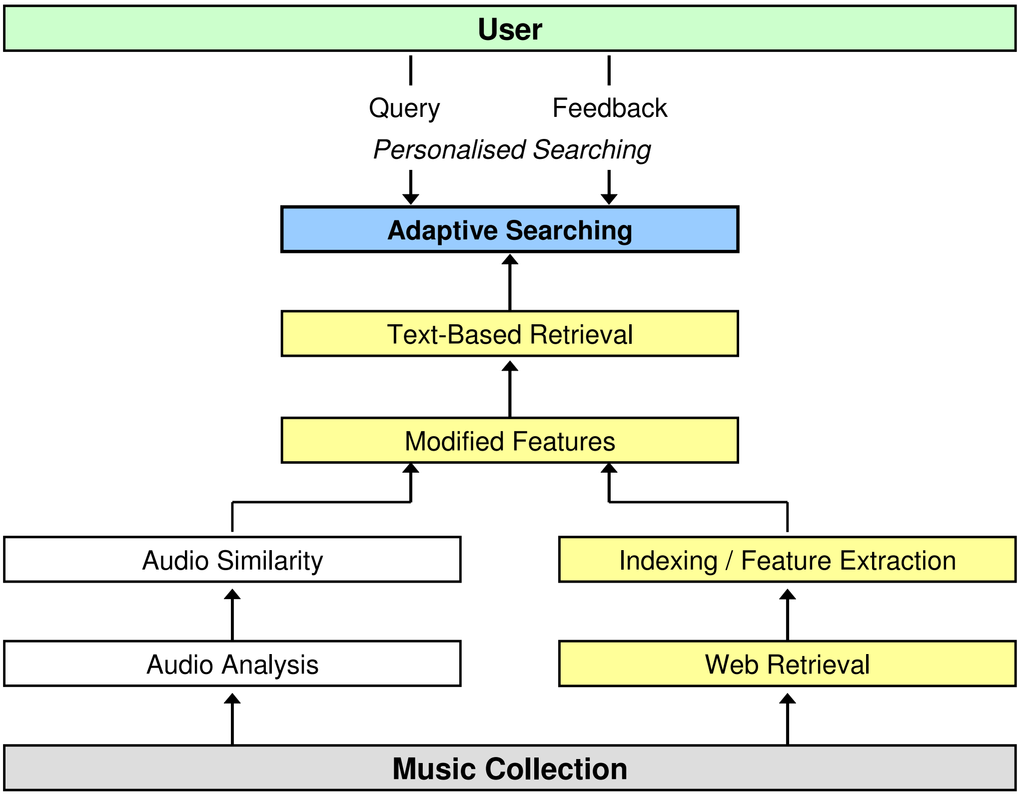


Figure 6.1: Schematic overview of the methods presented in this chapter. Yellow bars represent the proposed steps that lead to an adaptive search system (blue bar) that can exploit explicit feedback given by the user.

the user's preferences. Not only can the retrieval process increasingly accommodate to users expectations, the approach can also help to compensate for inadequately represented initial queries that would otherwise result in low performance. Note that the methodology used in this chapter is based on [Knees et al., 2007a] and is highly related to the pseudo document approach presented in Section 5.3.1. Since there exist some differences to the pseudo document approach, before introducing relevance feedback into the applied model, the processes of indexing and retrieval are briefly reviewed. A schematic overview of the methods presented in this chapter can be found in Figure 6.1.

6.2 Alternative Indexing and Retrieval

In contrast to the approaches presented in Chapter 5, in [Knees et al., 2007a] — apart from a different TF·IDF formulation — audio similarity is exploited to reduce the vector space dimensionality and to modify the term descriptors. For retrieval, query expansion is performed prior to comparison with the track descriptors. The following describes these steps briefly. For comparability, the same notation as in Chapter 5 is used.

Indexing using Term Selection and Audio Similarity

To calculate term profiles, for each music piece m and each term t appearing in I, $tf(t, m)$ (the term frequency of t in D_m) and $df(t, m)$ (the number of pages related to m in which the term t occurred) are counted. All terms with $df(t, m) \leq 2$ are removed from m's term set. Finally, $mpf(t)$ the number of music pieces that contain term t in their set is calculated (music piece frequency). Further, all terms that co-occur with less than 0.1% of all music pieces are removed. On the evaluation collection used for this task (cf. Section 6.4), this results in a vector space with about $78,000$ dimensions. To calculate the weight $w_{tfimpf}(t, m)$ of a term t for music piece m, a straight forward modification of the TF·IDF formulation as presented in Equation 3.2 is used.

$$w_{tfimpf}(t, m) = \begin{cases} [1 + \log_2 tf(t, m)] \log_2 \frac{|M|}{mpf(t)} & \text{if } tf(t, m) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

From the given definition, it can be seen that all Web pages related to a music piece are treated as one large document (cf. Section 5.3.1).

Dimensionality Reduction

For reducing the dimensionality of the feature space, the χ^2-test is applied. The χ^2-test is a standard term selection approach in text classification (e.g., [Yang and Pedersen, 1997]) and measures the independence of a term t from a given category or class c. Since in the task at hand there is no class information (e.g., genre information) available, MFCC-based audio similarity with proximity verification (see sections 3.2 and 3.2.2) is exploited instead. To this end, for each track m, a 2-class term selection problem is defined where the χ^2-test is used to find those terms that discriminate $N_{m,100}$, the group of the 100 most similar sounding tracks to m, from $U_{m,100}$, the group of the 100 most dissimilar sounding tracks to m. Thus, for each track

$$\chi^2(t, m) = \frac{N(AD - BC)^2}{(A + B)(A + C)(B + D)(C + D)} \tag{6.2}$$

is calculated, where A is the number of documents belonging to any piece in $N_{m,100}$ which contain term t (i.e., $\sum_{l \in N_{m,100}} df(t, l)$), B the number of documents belonging to any piece in $U_{m,100}$ which contain t, C the number of documents in $N_{m,100}$ without t, D the number of documents in $U_{m,100}$ without t, and N the total number of examined documents. For each track, the 50 terms with highest $\chi^2(t, m)$ values that occur more frequently in $N_{m,100}$ than in $U_{m,100}$ are joined into a global list. On the used evaluation collection, after this step, 4,679 distinct terms (feature dimensions) remain. Resulting term weight vectors are cosine normalised (cf. Equation 4.3) to remove the influence of the length of the retrieved Web pages as well as the different numbers of retrieved pages per track.

Vector Adaptation

Another use of the information provided by the audio similarity measure is the modification of the term vector representations toward acoustically similar pieces. This step is mandatory for tracks for which no related information could be retrieved from the Web. For all other tracks, the intention is to emphasise those dimensions that are typical among acoustically similar tracks. To this end, a simple Gauss weighting over the 10 most similar tracks is performed for each piece. Modified weights of term t for music piece m are defined as

$$w'_{tfimpf}(t,m) = \sum_{i=0}^{10} \frac{1}{\sqrt{2\pi}} e^{-\frac{(i/2)^2}{2}} \cdot w_{tfimpf}(t, a_i(m)), \tag{6.3}$$

Vectors are again cosine normalised after term weight adaptation.

Retrieval using Query Expansion

To be able to process a broad variety of queries, less sparse query vectors are obtained by performing query expansion. In the *on-line* version, the query is extended by the extra constraint *music* and sent to Google to construct a term vector from the 10 top Web pages returned. Alternatively, I, the *off-line* index of retrieved pages can be used for this operation. The resulting query vector $\vec{q}$ can then be compared to the music pieces in the collection by calculating Euclidean distances on the cosine normalised vectors (cf. Equation 4.4). From the distances, a relevance ranking is obtained which forms the response to the query.

6.3 Relevance Feedback

Relevance feedback is an iterative process in which the user is presented with a ranked list of the music pieces that are most similar to the query. After examination of the list, the user marks those pieces which are relevant in his/her opinion (*explicit relevance feedback*). In principle, implicit relevance feedback could also be deployed, e.g., by measuring the time a user is listening to the returned tracks. The intention is to modify the query vector such that it moves toward the relevant and away from the non-relevant pieces. Since both music pieces and queries are representable as weighted term vectors, the relevance feedback method by [Rocchio, 1971] can be easily incorporated to adapt search results according to users' preferences. Thus, based on the relevance judgements, the modified query vector $\vec{q}_{rf}$ can be calculated as

$$\vec{q}_{rf} = \alpha\, \vec{q} + \frac{\beta}{|D_r|} \sum_{\forall \vec{d}_i \in D_r} \vec{d}_i - \frac{\gamma}{|D_n|} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j \tag{6.4}$$

where $\vec{q}$ is the original query vector constructed from the returned Web pages, D_r the set of relevant music pieces (according to the user) among the retrieved pieces, and D_n the set of non-relevant pieces among the retrieved pieces (cf. [Baeza-Yates and Ribeiro-Neto, 1999], p.118-119). The parameters α, β, and γ can be used to tune the impacts of original vector, relevant pieces, and non-relevant pieces, respectively. In the presented experiments, equal values are assigned to all parameters, i.e., $\alpha = \beta = \gamma = 1$. The modified vector is again cosine normalised. Based on the new query vector, new results are presented to the user in the next step.

6.4 Evaluation

Table 6.1 gives an example that demonstrates the effects of relevance feedback. The left column shows the first 60 results for the query *Speed Metal* as returned by the text-based search engine. The right column shows the first 60 results if feedback is incorporated. Upon every 20 presented documents the query vector is modified based on the relevance information. Clearly, the number of relevant documents increases quickly after the first feedback iteration step.

6.4.1 Automatic Evaluation using Last.fm Tags

For automatic evaluation, the collection from [Knees et al., 2007a] is used. This collection comprises 12, 601 tracks by 1, 200 artists. Similar to the c35k collection presented in Section 5.6.1.1, Last.fm tags are used as test queries as well as for relevance indication (here in total 227 queries).

To measure the impact of relevance feedback (as well as of query expansion), for each test query three rankings are constructed. The first is the text-based ranking obtained via on-line query expansion. The second ranking is obtained by constructing a query vector via off-line query expansion. For the third ranking, relevance feedback is simulated by starting with the first 20 results obtained through the off-line based query vector. The next 20 results are then calculated from the query vector modified according to the relevance judgements of the already seen music pieces, and so on.

The precision at 11 standard recall level plots resulting for these three methods (averaged over all 227 test queries) are depicted in Figure 6.2. Not surprisingly, the usage of relevance feedback has a very positive effect on the precision of the returned music pieces. Starting from the same level (about 49% precision at recall level 0) the traditional approach without relevance feedback drops to 34% precision at recall level 10, while relevance feedback boosts precision to 52%. This trend is also clearly visible for all other recall levels. Besides this, it can be seen that the values of the off-line index approach without relevance feedback are constantly below the values of the on-line approach that uses Google for query vector construction.

Additional evaluation measures are shown in Table 6.2. Again, on-line query expansion using Google performs better than off-line expansion. For instance, for on-line expansion about 50% of the pieces among the first ten are relevant in average, whereas in average only 40% out of the first ten are relevant when using the off-line index. This number is consistent with the results obtained by means of a user study presented in the next section.

6.4.2 Evaluation via User Experiments

Additionally, a small user study with 11 participants has been conducted to gain insights into users' music search behaviour and to assess the impact of the relevance feedback under less artificial conditions. Each participant was asked to submit 5 queries of choice to the system. For each query, 5 feedback iterations with 20 results each were presented. Thus, in total, relevance to the query had to be judged for 100 results. Additionally, each query had to be evaluated twice. In one run, the ranking was not influenced by the ratings at all, i.e., the first 100 retrieval results without

	no relevance feedback	**with relevance feedback**
1.	Deicide - Dead But Dreaming	Deicide - Dead But Dreaming
2.	Deicide - Trifixion	Deicide - Trifixion
3.	Deicide - Repent To Die	Deicide - Repent To Die
4.	Skitzo - Kill With a Vengeance (live)	Skitzo - Kill With a Vengeance (live)
5.	Deicide - In Hell I Burn	Deicide - In Hell I Burn
6.	**Iron Savior - Protector**	**Iron Savior - Protector**
7.	Entombed - Chief Rebel Angel	Entombed - Chief Rebel Angel
8.	Deicide - Satan Spawn, The Caco-Daemon	Deicide - Satan Spawn, The Caco-Daemon
9.	**Iron Savior - Warrior**	**Iron Savior - Warrior**
10.	**Nightwish - Nightshade Forests**	**Nightwish - Nightshade Forests**
11.	**Powergod - Back To Attack**	**Powergod - Back To Attack**
12.	Deicide - Oblivious To Evil	Deicide - Oblivious To Evil
13.	**Steel Prophet - Unseen**	**Steel Prophet - Unseen**
14.	**Steel Prophet - The Ides Of March**	**Steel Prophet - The Ides Of March**
15.	**Steel Prophet - Messiah**	**Steel Prophet - Messiah**
16.	**Steel Prophet - Goddess Arise**	**Steel Prophet - Goddess Arise**
17.	**Steel Prophet - Ghosts Once Past**	**Steel Prophet - Ghosts Once Past**
18.	Deicide - Behead The Prophet	Deicide - Behead The Prophet
19.	Deicide - Revocate The Agitator	Deicide - Revocate The Agitator
20.	**Steel Prophet - Dawn Of Man**	**Steel Prophet - Dawn Of Man**
21.	**Steel Prophet - 07-03-47**	**Steel Prophet - 07-03-47**
22.	Deicide - Holy Deception	**Steel Prophet - Mysteries Of Inquity**
23.	**Steel Prophet - Mysteries Of Inquity**	**Powergod - Metal Church**
24.	Deicide - Sacrificial Suicide	**Powergod - Burning the Witches**
25.	**Powergod - Madhouse**	**Iron Savior - Paradise**
26.	Crematory - Lost In Myself - Trance Raymix	**Powergod - Madhouse**
27.	Tiamat - Cain	**Powergod - Bleed for the gods**
28.	**Powergod - Bleed for the gods**	**Iron Savior - For The World (Live)**
29.	**Powergod - Ruler Of The Wasteland**	**Iron Savior - Brave New World**
30.	**Powergod - Burning the Witches**	**Iron Savior - Mindfeeder**
31.	**Powergod - Metal Church**	**Powergod - Stars**
32.	Crematory - Through My Soul	**Powergod - Ruler Of The Wasteland**
33.	Crematory - Reign Of Fear	**Powergod - Esper**
34.	**Powergod - Soldiers Under Command**	**Stratovarius - Rebel**
35.	Tiamat - Carry Your Cross An Ill Carry...	**Powergod - Soldiers Under Command**
36.	**Powergod - Stars**	**Iron Savior - Crazy (Ltd Ed Bonus**
37.	Crematory - Revolution	**Iron Savior - Iron Savior (Live)**
38.	Crematory - Red Sky	Electric Six - She's White
39.	Entombed - Left Hand Path (Outro)	**Powergod - Salvation**
40.	Monoide - One year after first love	**Powergod - Prisoner**
41.	**Finntroll - Ursvamp**	**Powergod - The Eagle & The Rainbow**
42.	**Finntroll - Grottans Barn**	**Powergod - Anybody Home**
43.	**Powergod - Esper**	**Powergod - Lost Illusions**
44.	**Iron Savior - For The World (Live)**	**Powergod - Tor With The Hammer**
45.	**Finntroll - Fiskarens Fiende**	**Iron Savior - Riding On Fire (Live)**
46.	**Finntroll - Nattfodd**	**Powergod - Red Rum**
47.	**Finntroll - Trollhammaren**	**Powergod - Steel The Light**
48.	Chicks on Speed - Procrastinator	**Iron Savior - No Heroes**
49.	Deicide - Crucifixation	**Powergod - I Am A Viking**
50.	Entombed - Say It In Slugs	**Powergod - Into The Battle**
51.	**Iron Savior - Mindfeeder**	**Powergod - Kill With Power**
52.	Crematory - Dreams	**Powergod - Mean Clean Fighting Machine**
53.	Tiamat - Light In Extension	**Powergod - Children Of Lost Horizons**
54.	Deicide - Mephistopheles	**Powergod - I'm On Fire**
55.	**Iron Savior - Brave New World**	**Powergod - Gods Of War**
56.	Tiamat - Nihil	**Powergod - No Brain No Pain**
57.	**Iron Savior - Paradise**	**Powergod - Observator**
58.	Crematory - Human Blood	**Powergod - Evilution Part I**
59.	Entombed - Something Out Of Nothing	**Powergod - Powergod**
60.	**Stratovarius - Rebel**	Corvus Corax - Bitte Bitte

Table 6.1: Effects of relevance feedback on the query *Speed Metal*. Bold entries indicate relevant pieces according to Last.fm tag ground truth; automatic query update after 20 results.

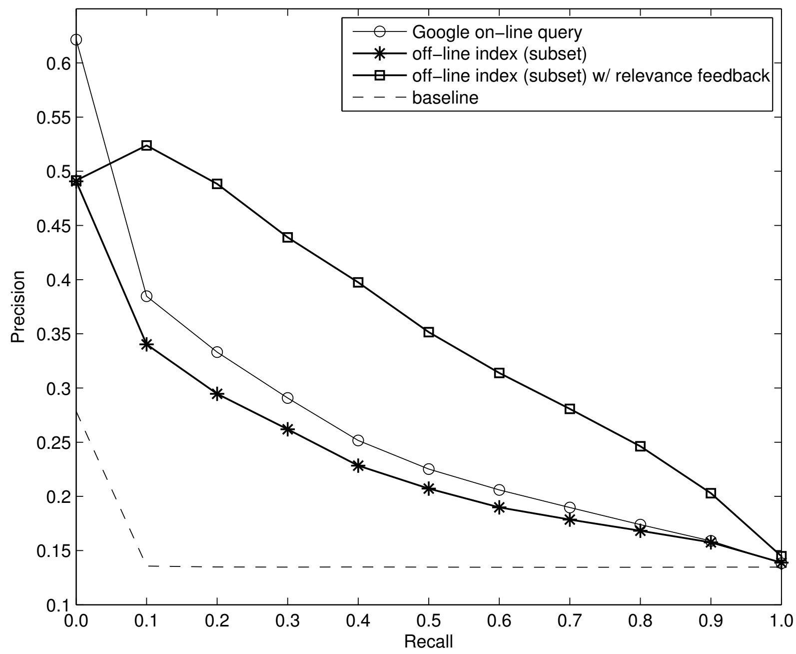


Figure 6.2: Precision at 11 standard recall level plots demonstrating the effects of relevance feedback (avg. over 227 queries).

	On-line	**Off-line**	**Feedback**
Prec@10	**49.56**	39.74	39.74
rPrec	26.41	23.48	**37.66**
AvgPrec	25.29	22.99	**35.80**

Table 6.2: IR measures showing the effect of relevance feedback (avg. over 227 test queries).

relevance feedback were presented in groups of 20. In the other run, relevance feedback was enabled. Thus, the ratings of the documents had a direct influence on the following 20 results. Whether the first or the second run was presented first was chosen randomly for each query to avoid learning effects. Furthermore, the users were told to evaluate two different feedback strategies. The fact that one run included no feedback strategy at all was concealed. The 55 different queries issued by the participants can be found in Table A.4.

Since obtaining users' relevance judgements for all pieces in the collection for all queries is infeasible, other measures than those used in Section 5.6.2 have to be applied to illustrate the impact of relevance feedback, such as the number of relevant pieces in each iteration step or the number of queries for which all results were considered relevant, cf. [Harman, 1992]. Table 6.3 displays the results of the user study. Interestingly, in the first iteration, results are not consistent. Obviously, users considered different tracks to be relevant in the first and in the second run (even if only very sporadically). Note that in the second iteration, the feedback

Iteration	**1**	**2**	**3**	**4**	**5**	**total**
			no relevance feedback			
Relevant retrieved/iter. (mean)	7.13	**5.02**	4.05	3.76	3.71	4.73
Relevant retrieved/iter. (sum)	392	**276**	223	207	204	1,302
Cumulative relevant retr. (sum)	392	**668**	891	1,098	1,302	1,302
Queries with all relevant	9	**3**	3	2	1	0
Queries with no relevant	22	**26**	25	28	30	17
			with relevance feedback			
Relevant retrieved/iter. (mean)	7.22	4.15	**6.47**	**5.73**	**6.18**	**5.95**
Relevant retrieved/iter. (sum)	397	228	**356**	**315**	**340**	**1,636**
Cumulative relevant retr. (sum)	397	625	**981**	**1,296**	**1,636**	**1,636**
Queries with all relevant	8	1	**6**	**4**	**5**	0
Queries with no relevant	23	27	**20**	**23**	**22**	**11**

Table 6.3: Results of the user study over 5 iterations. In each iteration, 55 queries were evaluated (the maximum achievable number of relevant retrieved pieces for each query is 20; the maximum achievable number per iteration is thus 1,100; advantageous values in bold typeface).

Category	**Queries**	**Relevant (avg.)**
Genre	28	33.25
Artist	12	28.50
Instrumentation	7	24.71
Track	6	2.50
Geographical	5	33.00
Movie related	4	16.00
Other	3	23.33
Total	55	29.75

Table 6.4: Identified query categories, the number of queries belonging to these categories, and the average number of relevant music pieces (out of 100).

approach performs worse due to an error during presentation, i.e., in the survey, the most relevant tracks after the first feedback step were not presented to the users and were therefore not included in the ratings. Nevertheless, the general trend of better results when using relevance feedback can still be observed.

From the set of issued queries, 6 basic types of queries can be identified.[1] Table 6.4 shows the different categories as well as the number of queries belonging to these categories. Note that a query can be assigned to multiple categories (e.g., *vienna electro dj* or *rammstein music with strong keyboard*). Worst results can be observed for queries that aim at finding a specific track. Although the user may select tracks other than the one specified, naturally the overall number of tracks rated relevant is very low. Furthermore, it can be seen that users are most satisfied with results for genre queries (e.g., *eurodance*) and geographically related queries (e.g., *new orleans*).

[1] Since searching for lyrics is currently not supported, queries addressing lyrics are not included.

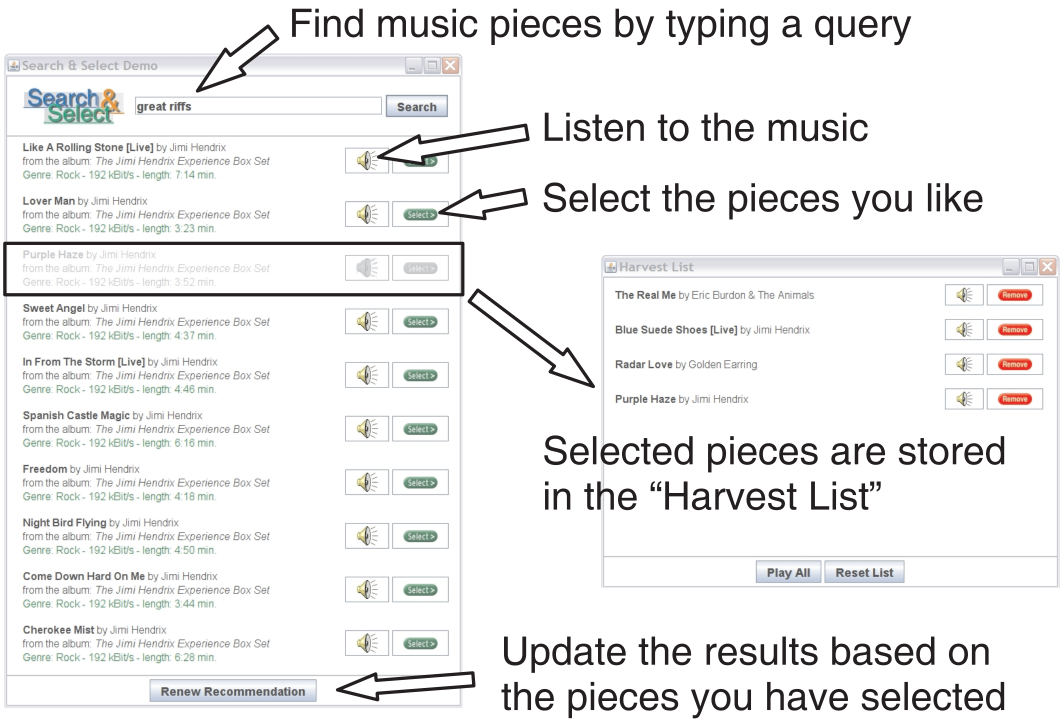


Figure 6.3: Overview of the Search'n'Select interface and usage instructions.

6.5 Prototype: The Search'n'Select Application

To demonstrate the applicability of the proposed methods for building interactive, user-centred search systems, a prototype called Search'n'Select has been developed, cf. [Knees, 2007]. Search'n'Select allows users to search for desired music in an iterative process that adapts to the user's behaviour. In the first step, the user can gain access to the underlying music collection by issuing simple free-form text queries. From the returned items, the user selects those after his/her fancy. The marked pieces are then transferred into a list of "harvested music pieces" (analogous to, e.g., a shopping cart in an on-line shop). Based on the chosen music pieces, the consecutively presented results are modified such that they tend to contain more pieces similar to the ones in the "harvest list". The user can continue searching by selecting (or ignoring) more results or by issuing the next query. Figure 6.3 shows an overview of the application and describes the process.

Search'n'Select can be seen as a combination of retrieval and browsing systems. By knowing which pieces are of interest to the user, the system can guide the user further in that direction by presenting other pieces that could be of interest. However, this guidance is very implicit, i.e., there is no (hierarchical) structure that informs the user about his/her current position within the collection. Therefore, a key aspect of this type of browsing is that potentially surprising results may be presented to the user and allow to explore further regions of the music space.

6.6 Recapitulation and Discussion

The approach presented in this chapter successfully incorporates relevance feedback into a search engine for large music collections that can be queried via natural language text input. Due to the underlying vector space model, Rocchio's method for including relevance feedback could be integrated smoothly. The conducted evaluations show that relevance feedback provides a valuable extension to the system in that it opens up the possibility to adapt to users' preferences. Furthermore, the Search'n'Select application prototype facilitates exploration of the music space by selecting pieces of interest and incorporates characteristics of both retrieval and browsing interfaces.

Apart from the improved results induced by relevance feedback, the positive impact on the system's performance allows to conclude that the vector space representations of the music pieces is well suited to model the similarity between pieces. To further advance the system, the translation of queries into the term vector space has to be improved. Starting with better initial results is also mandatory for the acceptance of the system since people usually judge the quality based on the first results.

Due to the similarity with the pseudo document approach introduced in Section 5.3.1, integration into this type of indexing poses no big problem. For the RRS-based retrieval approach from Section 5.3.2, however, a direct incorporation of the presented relevance feedback method is not possible due to the two-layered retrieval process. A possible workaround could consist in propagation of relevance judgements from music pieces back to the associated Web pages, where the query modification is performed then (cf. Section 5.4.2.1). From this modified query, a new music piece ranking could be calculated. Given these necessities, the pseudo document approach should be favoured when considering such extensions, as already concluded in Section 5.8.

Chapter 7

Critical Discussion and Future Directions

In this thesis, automatic methods to associate music pieces with textual descriptions extracted from the Web have been presented. The proposed techniques utilise common Web search engines to find related text content on the Web. From this content, descriptors are extracted and applied in three scenarios: to serve as labels that facilitate orientation within browsing interfaces to music collections, to be used as indexing terms for music retrieval systems that can be queried using descriptive free-form text as input, and as features in adaptive retrieval systems that aim at providing user-targeted music recommendations.

In general, exploiting Web-based texts permits to describe music and large music repositories with a diverse set of terms. One advantage of deriving textual features is that they are better understandable for humans than content-based descriptors. Labelling a piece or a region on a map with the words *Jazz* and *Piano* will (hopefully) better indicate to the typical user what type of music to expect than the presentation of sole statistics of spectral properties. However, this also requires having some background on musical terminology. As with genre descriptors, not everybody may be familiar with words that describe finer grained music concepts. Therefore, it may be valuable to have some redundancy within the applied descriptors, i.e., synonyms or closely related concepts (something that seems to emerge naturally in collaborative tagging systems, cf. Figure 2.2). Multiple labels also allow for emphasis of specific aspects of the music and to clarify ambiguous descriptors (i.e., polysemous music descriptors like the term *Indian music*, cf. Chapter 6.1)).

The most severe problem with this type of representation is that in order to be able to describe music with related terms, these terms have to be available. Not only must a music piece have gained a certain popularity to be present in the Web at all; even if Web-data is available and ranked accordingly by the used Web search engine, there is no guarantee that its contents deal with the "right" topics or make use of the "right" vocabulary. That is, the presented approaches are rather fragile in that they rely on the existence and accessibility of specific expressions in specific contexts. When using the descriptors for labelling — as done in the Music Description Map technique and in the nepTune interface — this drawback might be acceptable since similar descriptions could be available instead to assist the user. When using the descriptions as features, however, this has severe implications. In the proposed feature space, every term corresponds to a dimension. These dimensions are considered independent (which is in general not true) and equally important (which is relevant when calculating similarities, for instance). Hence, words of interest not present on Web pages affect the indexing heavily and result in features that have low quality and may be a bad representation for music tracks.

A possible approach to deal with this is to abandon the idea of the bag-of-words vector space model and follow a more stable IR approach, i.e., a semantic indexing approach. By exploiting such an approach that makes use of dimensionality reduction methods such as LSA [Deerwester et al., 1990] or NMF [Lee and Seung, 1999], instead of working on a direct term-document representation, a set of "semantic" concepts that consist of a weighted combination of words is produced and documents are represented within the resulting concept space. Although weighted combinations of terms may be less intuitive, the advantages are manifold. First, the resulting concept representation has lower dimensionality since original term vector dimensions corresponding to terms that have similar meanings are merged into a joint concept dimension. In the resulting space, polysemous words can be identified since they contribute to multiple concepts, i.e., they can be observed in different "semantic" contexts simultaneously. Second, noisy term dimensions are omitted or marginalised since they do not contribute to a specific concept. Third, resulting document representations are less sparse. Initial steps that apply such a method to music-related Web terms have been made by [Pohle et al., 2007b]. To make the methods presented in this thesis more robust and therefore improve retrieval quality in corresponding applications, a similar strategy could be pursued.

Another aspect tackled in this thesis is the combination of Web-based descriptors with acoustic similarity extracted from the audio signal. While complementing primarily audio-based techniques with Web data has shown good potential (as in the MDM and the nepTune interface), incorporation of audio-based similarity into text-based retrieval as proposed in this thesis was less successful. However, it is likely that other forms of combination would yield better results. First, as has been demonstrated by [Barrington et al., 2009], supervised learning of a pre-specified set of music-relevant concepts is feasible. A combination of a similar strategy with the concept-based text descriptions mentioned before is conceivable for future work. Alternatively, for text-based retrieval, a more sophisticated combination with audio similarity could be realised, e.g., by integrating (Markov chain) random walks on the audio similarity graph starting from a text-based music piece ranking.

For future research, another very interesting contribution and addition to the work carried out so far would be the development of an autonomous Web crawler to index the Web and to replace the functionality and position currently filled by commercial Web search engines. To make such a task feasible, this focused crawler would have to be trained using machine learning techniques to identify domain specific Web pages, to follow promising links with higher priority, and, hence, to index only those parts of the Web that are relevant for subsequent tasks such as those discussed in this thesis (cf. Section 2.2.1). For building such an application, previously developed techniques to extract music-relevant information, such as estimation of artist similarity, co-occurrence analysis, and prototypical artist detection, can be reused.

In addition to Web-based indexing and retrieval, new application areas for Web data in the music domain seem very interesting — and all of these would also benefit from having full access to a (music targeted) Web index. For instance, approaches for entity detection to discover, for instance, new artists could be applied. This would be a straight forward extension to existing co-occurrence techniques, capable of revealing new information. With this kind of technology, new techniques for discovery of emerging trends in music are conceivable. Another potential research direction that goes beyond the implicit information modelling contained in the current context-based indexing (such as implicit band-member relations, cf. Figure 5.9) is music information extraction. For instance, having a collection of music-specific Web pages, explicit information about artists, albums or individual pieces can be extracted. By applying predefined rules (or rules automatically derived from labelled examples) these pages can be examined for meta-information such as discographies, biographical data of artists (or bands, e.g., the history of band members), or relations between artists, composers, or pieces. For instance, such a relation could indicate whether a song is a cover version of another. If this information is not explicitly available, other information present from the Web could be used, e.g., for determining the composer of a given piece or by comparing the lyrics. Furthermore, such an approach has the potential to be combined with audio-based cover version detection approaches to include additional evidence. In general, instead of modelling music pieces or artists by weightings of terms or concepts, the development of intelligent methods to derive factual information about music entities and their connections is very interesting. Such explicit meta-data is not only a valuable source for structuring repositories and for making recommendations and therefore of interest for digital music resellers, but also useful for complementing existing intelligent context- and content-based applications (e.g., for constraining uninformed recommendation systems and to avoid implausible or inappropriate suggestions such as *Christmas music* outside the holiday season).

In conclusion of this thesis, it can be stated that Web-based indexing and retrieval of music (as well as Web-based MIR in general) is a very important and relevant research direction. Special attention should be given to the fact that Web-MIR methods are capable of representing aspects of music not encoded in the audio signal. Thus, finding the right balance between content- and context-based methods appears to be the key for the development of future music applications.

Bibliography

[Alani et al., 2003] Alani, H., Kim, S., Millard, D. E., Weal, M. J., Hall, W., Lewis, P. H., and Shadbolt, N. R. (2003). Automatic Ontology-Based Knowledge Extraction from Web Documents. *IEEE Intelligent Systems*, 18(1):14–21.

[Anderson, 2006] Anderson, C. (2006). *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, New York.

[Anglade et al., 2007] Anglade, A., Tiemann, M., and Vignoli, F. (2007). Complex-Network Theoretic Clustering for Identifying Groups of Similar Listeners in P2P Systems. In *Proceedings of the 1st ACM International Conference on Recommender Systems (RecSys'07)*, pages 41–48, Minneapolis, Minnesota, USA.

[Aucouturier, 2006] Aucouturier, J.-J. (2006). *Ten Experiments on the Modelling of Polyphonic Timbre*. PhD thesis, University of Paris 6.

[Aucouturier and Pachet, 2002a] Aucouturier, J.-J. and Pachet, F. (2002a). Finding songs that sound the same. In *Proceedings of the IEEE Benelux Workshop on Model Based Processing and Coding of Audio*, pages 91–98, Lueven, Belgium.

[Aucouturier and Pachet, 2002b] Aucouturier, J.-J. and Pachet, F. (2002b). Music Similarity Measures: What's the Use? In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR'02)*, pages 157–163, Paris, France.

[Aucouturier and Pachet, 2002c] Aucouturier, J.-J. and Pachet, F. (2002c). Scaling Up Music Playlist Generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'02)*, pages 105–108, Lausanne, Switzerland.

[Aucouturier and Pachet, 2003] Aucouturier, J.-J. and Pachet, F. (2003). Musical Genre: A Survey. *Journal of New Music Research*, 32(1).

[Aucouturier and Pachet, 2004] Aucouturier, J.-J. and Pachet, F. (2004). Improving Timbre Similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1).

[Aucouturier et al., 2007] Aucouturier, J.-J., Pachet, F., Roy, P., and Beurivé, A. (2007). Signal + Context = Better Classification. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Baccigalupo et al., 2008] Baccigalupo, C., Plaza, E., and Donaldson, J. (2008). Uncovering Affinity of Artists to Multiple Genres from Social Behaviour Data. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, Philadelphia, PA, USA.

[Baeza-Yates and Ribeiro-Neto, 1999] Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley, Reading, Massachusetts.

[Barrington et al., 2009] Barrington, L., Turnbull, D., Yazdani, M., and Lanckriet, G. (2009). Combining Audio Content and Social Context for Semantic Music Discovery. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09)*, Boston, MA, USA.

[Baumann, 2005] Baumann, S. (2005). *Artificial Listening Systems: Modellierung und Approximation der subjektiven Perzeption von Musikähnlichkeit*. PhD thesis, Technical University of Kaiserslautern.

[Baumann and Hummel, 2003] Baumann, S. and Hummel, O. (2003). Using Cultural Metadata for Artist Recommendation. In *Proceedings of the 3rd International Conference on Web Delivering of Music (WEDELMUSIC 2003)*.

[Baumann et al., 2002] Baumann, S., Klüter, A., and Norlien, M. (2002). Using natural language input and audio analysis for a human-oriented MIR system. In *Proceedings of the 2nd International Conference on Web Delivering of Music (WEDELMUSIC 2002)*, Darmstadt, Germany.

[Baur et al., 2010] Baur, D., Steinmayr, B., and Butz, A. (2010). SongWords: Exploring Music Collections Through Lyrics. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR'10)*, Utrecht, Netherlands.

[Bello and Pickens, 2005] Bello, J. P. and Pickens, J. (2005). A robust mid-level representation for harmonic content in music signals. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK.

[Berenzweig et al., 2003] Berenzweig, A., Logan, B., Ellis, D. P., and Whitman, B. (2003). A Large-Scale Evaluation of Acoustic and Subjective Music Similarity Measures. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR'03)*, Baltimore, Maryland.

[Bishop, 1995] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[Brill, 1992] Brill, E. (1992). A Simple Rule-Based Part of Speech Tagger. In *Proceedings of the 3rd Conference on Applied Natural Language Processing*, pages 152–155.

[Brochu et al., 2003] Brochu, E., de Freitas, N., and Bao, K. (2003). The Sound of an Album Cover: Probabilistic Multimedia and IR. In *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*, Key West, Florida, USA.

[Byrd and Fingerhut, 2002] Byrd, D. and Fingerhut, M. (2002). The History of ISMIR A Short Happy Tale. *D-Lib Magazine*, 8(11).

[Cai et al., 2003] Cai, D., Yu, S., Wen, J.-R., and Ma, W.-Y. (2003). Extracting Content Structure for Web Pages based on Visual Representation. In *Proceedings of the 5th Asia Pacific Web Conference (APWeb'03)*.

[Cano et al., 2002] Cano, P., Kaltenbrunner, M., Gouyon, F., and Batlle, E. (2002). On the Use of Fastmap for Audio Retrieval and Browsing. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR'02)*, pages 275–276, Paris, France.

[Cano and Koppenberger, 2004] Cano, P. and Koppenberger, M. (2004). The Emergence of Complex Network Patterns in Music Artist Networks. In *Proceedings of the 5th International Symposium on Music Information Retrieval (ISMIR'04)*, pages 466–469, Barcelona, Spain.

[Casey et al., 2008] Casey, M. A., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., and Slaney, M. (2008). Content-Based Music Information Retrieval: Current Directions and Future Challenges. *Proceedings of the IEEE*, 96:668–696.

[Celino et al., 2006] Celino, I., Della Valle, E., Cerizza, D., and Turati, A. (2006). Squiggle: a Semantic Search Engine for Indexing and Retrieval of Multimedia Content. In *Proceedings of the 1st International Workshop on Semantic-enhanced Multimedia Presentation Systems (SEMPS'06)*, Athens, Greece.

[Celma, 2008] Celma, O. (2008). *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain.

[Celma et al., 2006] Celma, O., Cano, P., and Herrera, P. (2006). Search Sounds: An audio crawler focused on weblogs. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR'06)*, Victoria, B.C., Canada.

[Celma and Lamere, 2007] Celma, O. and Lamere, P. (2007). ISMIR 2007 Tutorial: Music Recommendation.
URL: `http://mtg.upf.edu/~ocelma/MusicRecommendationTutorial-ISMIR2007/`
[Online; accessed 14-Nov-2010].

[Celma and Nunes, 2008] Celma, O. and Nunes, M. (2008). GeoMuzik: A geographic interface for large music collections. In *ISMIR 2008: 9th International Conference on Music Information Retrieval – Late-Breaking/Demo Session*.

[Chakrabarti et al., 1999] Chakrabarti, S., van den Berg, M., and Dom, B. (1999). Focused Crawling: A New Approach to Topic-specific Web Resource Discovery. *Computer Networks*, 31(11–16):1623–1640.

[Cohen and Fan, 2000] Cohen, W. and Fan, W. (2000). Web-Collaborative Filtering: Recommending Music by Crawling The Web. *WWW9 / Computer Networks*, 33(1-6):685–698.

[Corpet, 1988] Corpet, F. (1988). Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research*, 16(22):10881–10890.

[Cover and Hart, 1967] Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

[Cox and Cox, 2000] Cox, T. F. and Cox, M. A. A. (2000). *Multidimensional Scaling*. Cox, T. F., Cox, M. A. A., and Cox, T. F. (2000). Multidimensional Scaling, Second Edition. Chapman & Hall/CRC., 2nd edition edition.

[Cunningham et al., 2002] Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2002). GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia.

[Cunningham et al., 2004] Cunningham, S. J., Jones, M., and Jones, S. (2004). Organizing Digital Music for Use: An Examination of Personal Music Collections. In *Proceedings of 5th International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain.

[Debnath et al., 2005] Debnath, S., Mitra, P., and Giles, C. L. (2005). Automatic extraction of informative blocks from webpages. In *Proceedings of the 20th ACM Symposium on Applied Computing (SAC'05)*.

[Debole and Sebastiani, 2003] Debole, F. and Sebastiani, F. (2003). Supervised term weighting for automated text categorization. In *Proceedings of the 18th ACM Symposium on Applied Computing (SAC'03)*, pages 784–788, Melbourne, US.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

[Diakopoulos et al., 2009] Diakopoulos, D., Vallis, O., Hochenbaum, J., Murphy, J., and Kapur, A. (2009). 21st Century Electronica: MIR Techniques for Classification and Performance. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09)*, Kobe, Japan.

[Donaldson, 2007] Donaldson, J. (2007). A Hybrid Social-Acoustic Recommendation System for Popular Music. In *Proceedings of the 1st ACM International Conference on Recommender Systems (RecSys'07)*, pages 187–190, Minneapolis, Minnesota, USA.

[Eck et al., 2007] Eck, D., Bertin-Mahieux, T., and Lamere, P. (2007). Autotagging Music Using Supervised Machine Learning. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Eck et al., 2008] Eck, D., Lamere, P., Bertin-Mahieux, T., and Green, S. (2008). Automatic Generation of Social Tags for Music Recommendation. In *Advances in Neural Information Processing Systems 20 (NIPS'07)*. MIT Press.

[Ellis et al., 2002] Ellis, D., Whitman, B., Berenzweig, A., and Lawrence, S. (2002). The Quest For Ground Truth in Musical Artist Similarity. In *Proceedings of 3rd International Conference on Music Information Retrieval (ISMIR'02)*, Paris, France.

[Fields et al., 2008] Fields, B., Jacobson, K., Rhodes, C., and Casey, M. (2008). Social Playlists and Bottleneck Measurements: Exploiting Musician Social Graphs Using Content-Based Dissimilarity and Pairwise Maximum Flow Values. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, Philadelphia, PA, USA.

[Flexer, 2007] Flexer, A. (2007). A closer look on artist filters for musical genre classification. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Flexer et al., 2008] Flexer, A., Schnitzer, D., Gasser, M., and Widmer, G. (2008). Playlist generation using start and end songs. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*.

[Foote, 1997] Foote, J. T. (1997). Content-Based Retrieval of Music and Audio. In Kuo, C., editor, *Proceedings of SPIE Multimedia Storage and Archiving Systems II*, volume 3229, pages 138–147, Bellingham, WA.

[Frankel et al., 1996] Frankel, C., Swain, M. J., and Athitsos, V. (1996). Webseer: An image search engine for the world wide web. Technical Report TR-96-14, University of Chicago, Department of Computer Science.

[Gasser and Flexer, 2009] Gasser, M. and Flexer, A. (2009). FM4 Soundpark: Audio-based Music Recommendation in Everyday Use. In *Proceedings of the 6th Sound and Music Computing Conference (SMC'09)*, Porto, Portugal.

[Geleijnse and Korst, 2006] Geleijnse, G. and Korst, J. (2006). Web-based artist categorization. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR'06)*, Victoria, Canada.

[Geleijnse and Korst, 2007] Geleijnse, G. and Korst, J. (2007). Tool Play Live: Dealing with Ambiguity in Artist Similarity Mining from the Web. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Geleijnse et al., 2007] Geleijnse, G., Schedl, M., and Knees, P. (2007). The Quest for Ground Truth in Musical Artist Tagging in the Social Web Era. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Ghias et al., 1995] Ghias, A., Logan, J., Chamberlin, D., and Smith, B. C. (1995). Query by Humming: Musical Information Retrieval in an Audio Database. In

Proceedings of the 3rd ACM International Conference on Multimedia, pages 231–236, San Fancisco, CA, USA.

[Gospodnetić and Hatcher, 2005] Gospodnetić, O. and Hatcher, E. (2005). *Lucene in Action*. Manning.

[Goto and Goto, 2005] Goto, M. and Goto, T. (2005). Musicream: New Music Playback Interface for Streaming, Sticking, and Recalling Musical Pieces. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK.

[Govaerts and Duval, 2009] Govaerts, S. and Duval, E. (2009). A Web-Based Approach to Determine the Origin of an Artist. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09)*, Kobe, Japan.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explorations*, 11(1).

[Harman, 1992] Harman, D. (1992). Relevance Feedback Revisited. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92)*, Copenhagen, Denmark.

[Herscovici et al., 2007] Herscovici, M., Lempel, R., and Yogev, S. (2007). Efficient indexing of versioned document sequences. In *Proceedings of the 29th European Conference on Information Retrieval (ECIR'07)*, pages 76–87, Rome, Italy.

[Hirjee and Brown, 2009] Hirjee, H. and Brown, D. G. (2009). Automatic Detection of Internal and Imperfect Rhymes in Rap Lyrics. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09)*, Kobe, Japan.

[Hirjee and Brown, 2010] Hirjee, H. and Brown, D. G. (2010). Rhyme Analyzer: An Analysis Tool for Rap Lyrics. In *ISMIR 2010: 11th International Society for Music Information Retrieval Conference – Late-Breaking/Demo Session*, Utrecht, Netherlands.

[Hitchner et al., 2007] Hitchner, S., Murdoch, J., and Tzanetakis, G. (2007). Music Browsing Using a Tabletop Display. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Hofmann, 1999] Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence (UAI'99)*, Stockholm.

[Hollander and Wolfe, 1999] Hollander, M. and Wolfe, D. A. (1999). *Nonparametric Statistical Methods*. Wiley-Interscience, 2nd edition edition.

[Hu et al., 2009] Hu, X., Downie, J. S., and Ehmann, A. F. (2009). Lyric Text Mining in Music Mood Classification. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09)*, Kobe, Japan.

[Hu et al., 2005] Hu, X., Downie, J. S., West, K., and Ehmann, A. (2005). Mining Music Reviews: Promising Preliminary Results. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK.

[Jacobson et al., 2008] Jacobson, K., Fields, B., and Sandler, M. (2008). Using Audio Analysis and Network Structure to Identify Communities in On-Line Social Networks of Artists. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, Philadelphia, PA, USA.

[Joachims, 1998] Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the European Conference on Machine Learning (ECML'98)*, pages 137–142, Berlin, Germany.

[Julià and Jordà, 2009] Julià, C. F. and Jordà, S. (2009). SongExplorer: A Tabletop Application for Exploring Large Collections of Songs. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09)*, Kobe, Japan.

[Kaski et al., 1998] Kaski, S., Honkela, T., Lagus, K., and Kohonen, T. (1998). WEBSOM – Self-Organizing Maps of Document Collections. *Neurocomputing*, 21:101–117.

[Kaski et al., 1999] Kaski, S., Venna, J., and Kohonen, T. (1999). Coloring That Reveals High-Dimensional Structures in Data. In Gedeon, T., Wong, P., Halgamuge, S., Kasabov, N., Nauck, D., and Fukushima, K., editors, *Proceedings of the 6th International Conference on Neural Information Processing (ICONIP'99)*, volume II, pages 729–734.

[Kherfi et al., 2004] Kherfi, M. L., Ziou, D., and Bernardi, A. (2004). Image Retrieval from the World Wide Web: Issues, Techniques, and Systems. *ACM Computing Surveys*, 36(1):35–67.

[Kim et al., 2009] Kim, J. H., Tomasik, B., and Turnbull, D. (2009). Using Artist Similarity to Propagate Semantic Information. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09)*, Kobe, Japan.

[Kleedorfer et al., 2008] Kleedorfer, F., Knees, P., and Pohle, T. (2008). Oh Oh Oh Whoah! Towards Automatic Topic Detection In Song Lyrics. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, pages 287 – 292, Philadelphia, PA, USA.

[Kleinberg, 1999] Kleinberg, J. M. (1999). Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632.

[Knees, 2007] Knees, P. (2007). Search & Select - Intuitively Retrieving Music from Large Collections. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Knees et al., 2004] Knees, P., Pampalk, E., and Widmer, G. (2004). Artist Classification with Web-based Data. In *Proceedings of 5th International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain.

[Knees et al., 2008a] Knees, P., Pohle, T., Schedl, M., Schnitzer, D., and Seyerlehner, K. (2008a). A Document-centered Approach to a Natural Language Music Search Engine. In *Proceedings of the 30th European Conference on Information Retrieval (ECIR'08)*, Glasgow, Scotland, UK.

[Knees et al., 2009] Knees, P., Pohle, T., Schedl, M., Schnitzer, D., Seyerlehner, K., and Widmer, G. (2009). Augmenting Text-Based Music Retrieval with Audio Similarity. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09)*, Kobe, Japan.

[Knees et al., 2006a] Knees, P., Pohle, T., Schedl, M., and Widmer, G. (2006a). Combining Audio-based Similarity with Web-based Data to Accelerate Automatic Music Playlist Generation. In *Proceedings of the 8th ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR'06)*, Santa Barbara, California, USA.

[Knees et al., 2007a] Knees, P., Pohle, T., Schedl, M., and Widmer, G. (2007a). A Music Search Engine Built upon Audio-based and Web-based Similarity Measures. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*, Amsterdam, Netherlands.

[Knees et al., 2007b] Knees, P., Pohle, T., Schedl, M., and Widmer, G. (2007b). Exploring Music Collections in Virtual Landscapes. *IEEE MultiMedia*, 14(3):46–54.

[Knees et al., 2008b] Knees, P., Schedl, M., and Pohle, T. (2008b). A Deeper Look into Web-based Classification of Music Artists. In *Proceedings of 2nd Workshop on Learning the Semantics of Audio Signals (LSAS'08)*, Paris, France.

[Knees et al., 2010] Knees, P., Schedl, M., Pohle, T., Seyerlehner, K., and Widmer, G. (2010). Supervised and Unsupervised Web Document Filtering Techniques to Improve Text-Based Music Retrieval. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR'10)*, Utrecht, Netherlands.

[Knees et al., 2006b] Knees, P., Schedl, M., Pohle, T., and Widmer, G. (2006b). An Innovative Three-Dimensional User Interface for Exploring Music Collections Enriched with Meta-Information from the Web. In *Proceedings of the 14th ACM International Conference on Multimedia*, Santa Barbara, California, USA.

[Knees et al., 2005] Knees, P., Schedl, M., and Widmer, G. (2005). Multiple Lyrics Alignment: Automatic Retrieval of Song Lyrics. In *Proceedings of 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK.

[Knopke, 2004] Knopke, I. (2004). AROOOGA: An audio search engine for the World Wide Web. In *Proceedings of the 2004 International Computer Music Conference (ICMC'04)*, Miami, USA.

[Kohonen, 2001] Kohonen, T. (2001). *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, 3rd edition.

[Kornstädt, 1998] Kornstädt, A. (1998). Themefinder: A Web-based Melodic Search Tool. *Computing in Musicology*, 11:231–236.

[Korst and Geleijnse, 2006] Korst, J. and Geleijnse, G. (2006). Efficient lyrics retrieval and alignment. In Verhaegh, W., Aarts, E., ten Kate, W., Korst, J., and Pauws, S., editors, *Proceedings of the 3rd Philips Symposium on Intelligent Algorithms (SOIA'06)*, pages 205–218, Eindhoven, the Netherlands.

[Krenmair, 2008] Krenmair, A. (2008). Integration einer Playlistfunktionalität in nepTune. Bakkalaureatsarbeit, Department of Computational Perception, Johannes Kepler Universität, Linz, Austria.

[Krenmair, 2010] Krenmair, A. (2010). Musikspezifische Informationsextraktion aus Webdokumenten. Diplomarbeit, Johannes Kepler University, Linz, Austria.

[Lagus and Kaski, 1999] Lagus, K. and Kaski, S. (1999). Keyword Selection Method for Characterizing Text Document Maps. In *Proceedings of the 9th International Conference on Artificial Neural Networks (ICANN'99)*, volume 1, pages 371–376, London.

[Lamere and Eck, 2007] Lamere, P. and Eck, D. (2007). Using 3D Visualizations to Explore and Discover Music. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*.

[Laurier et al., 2008] Laurier, C., Grivolla, J., and Herrera, P. (2008). Multimodal Music Mood Classification using Audio and Lyrics. In *Proceedings of the International Conference on Machine Learning and Applications (ICMLA'02)*, San Diego, CA, USA.

[Laurier et al., 2009] Laurier, C., Sordo, M., Serrà, J., and Herrera, P. (2009). Music Mood Representations from Social Tags. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09)*, Kobe, Japan.

[Law et al., 2007] Law, E., von Ahn, L., Dannenberg, R., and Crawford, M. (2007). Tagatune: A Game for Music and Sound Annotation. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Lee and Seung, 1999] Lee, D. D. and Seung, H. S. (1999). Learning the Parts of Objects by Non-negative Matrix Factorization. *Nature*, 401(6755):788–791.

[Leitich and Topf, 2007] Leitich, S. and Topf, M. (2007). Globe of Music - Music Library Visualization using GeoSOM. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Levy and Sandler, 2008] Levy, M. and Sandler, M. (2008). Learning latent semantic models for music from social tags. *Journal of New Music Research*, 37(2):137–150.

[Lew et al., 2006] Lew, M. S., Sebe, N., Djeraba, C., and Jain, R. (2006). Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2(1):1–19.

[Liaw and Wiener, 2002] Liaw, A. and Wiener, M. (2002). Classification and Regression by randomForest. *R News*, 2/3:18–22.

[Lībeks and Turnbull, 2010] Lībeks, J. and Turnbull, D. (2010). Exploring 'Artist Image' Using Content-Based Analysis of Promotional Photos. In *Proceedings of the International Computer Music Conference (ICMC) 2010*, New York, NY, USA.

[Lillie, 2008] Lillie, A. (2008). MusicBox: Navigating the space of your music. Masters thesis, Massachusetts Institute of Technology.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 4(1).

[Liu, 2007] Liu, B. (2007). *Web Data Mining – Exploring Hyperlinks, Contents and Usage Data*. Springer, Berlin, Heidelberg, Germany.

[Logan, 2000] Logan, B. (2000). Mel Frequency Cepstral Coefficients for Music Modeling. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR'00)*, Plymouth, MA, USA.

[Logan et al., 2003] Logan, B., Ellis, D., and Berenzweig, A. (2003). Toward evaluation techniques for music similarity. In *Keynote address, Workshop on the Evaluation of Music Information Retrieval (MIR) Systems at SIGIR*.

[Logan et al., 2004] Logan, B., Kositsky, A., and Moreno, P. (2004). Semantic Analysis of Song Lyrics. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'04)*, Taipei, Taiwan.

[Logan and Salomon, 2001] Logan, B. and Salomon, A. (2001). A Music Similarity Function Based on Signal Analysis. In *P*roceedings of the IEEE International Conference on Multimedia and Expo (ICME'01), Tokyo, Japan.

[Lübbers and Jarke, 2009] Lübbers, D. and Jarke, M. (2009). Adaptive Multimodal Exploration of Music Collections. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09)*, pages 195–200, Kobe, Japan.

[Maddage et al., 2006] Maddage, N. C., Li, H., and Kankanhalli, M. S. (2006). Music Structure Based Vector Space Retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)*, Seattle, WA, USA.

[Mahedero et al., 2005] Mahedero, J. P. G., Martínez, A., Cano, P., Koppenberger, M., and Gouyon, F. (2005). Natural language processing of lyrics. In *Proceedings of the 13th ACM International Conference on Multimedia*, pages 475–478, Singapore.

[Mandel and Ellis, 2005] Mandel, M. and Ellis, D. (2005). Song-Level Features and Support Vector Machines for Music Classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK.

[Mandel and Ellis, 2007] Mandel, M. and Ellis, D. (2007). A Web-based Game for Collecting Music Metadata. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Marolt, 2006] Marolt, M. (2006). A mid-level melody-based representation for calculating audio similarity. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR'06)*, Victoria, BC, Canada.

[Mayer et al., 2008] Mayer, R., Neumayer, R., and Rauber, A. (2008). Rhyme and Style Features for Musical Genre Classification by Song Lyrics. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, Philadelphia, PA, USA.

[McKay and Fujinaga, 2008] McKay, C. and Fujinaga, I. (2008). Combining Features Extracted From Audio, Symbolic and Cultural Sources. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, pages 597–602, Philadelphia, PA, USA.

[Mörchen et al., 2005] Mörchen, F., Ultsch, A., Nöcker, M., and Stamm, C. (2005). Databionic visualization of music collections according to perceptual distance. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK.

[Needleman and Wunsch, 1970] Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453.

[Neumayer et al., 2005] Neumayer, R., Dittenbach, M., and Rauber, A. (2005). PlaySOM and PocketSOMPlayer, Alternative Interfaces to Large Music Collections. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK.

[Pachet and Cazaly, 2000] Pachet, F. and Cazaly, D. (2000). A Taxonomy of Musical Genres. In *Proceedings of 6th International RIAO Conference on Content-Based Multimedia Information Access*, pages 1238–1245, Paris, France.

[Pachet et al., 2001] Pachet, F., Westerman, G., and Laigre, D. (2001). Musical Data Mining for Electronic Music Distribution. In *Proceedings of the 1st International Conference on Web Delivering of Music (WEDELMUSIC 2001)*, Florence, Italy.

[Page et al., 1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab.

[Pampalk, 2001] Pampalk, E. (2001). Islands of Music: Analysis, Organization, and Visualization of Music Archives. Master's thesis, Vienna University of Technology.

[Pampalk, 2006] Pampalk, E. (2006). *Computational Models of Music Similarity and their Application to Music Information Retrieval*. PhD thesis, Vienna University of Technology, Austria.

[Pampalk et al., 2004] Pampalk, E., Dixon, S., and Widmer, G. (2004). Exploring Music Collections by Browsing Different Views. *Computer Music Journal*, 28(2):49–62.

[Pampalk et al., 2005] Pampalk, E., Flexer, A., and Widmer, G. (2005). Hierarchical Organization and Description of Music Collections at the Artist Level. In *Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL'05)*, pages 37–48, Vienna, Austria.

[Pampalk and Goto, 2006] Pampalk, E. and Goto, M. (2006). MusicRainbow: A New User Interface to Discover Artists Using Audio-based Similarity and Web-based Labeling. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR'06)*, Victoria, B.C., Canada.

[Pampalk and Goto, 2007] Pampalk, E. and Goto, M. (2007). MusicSun: A New Approach to Artist Recommendation. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Pampalk et al., 2002] Pampalk, E., Rauber, A., and Merkl, D. (2002). Content-Based Organization and Visualization of Music Archives. In *Proceedings of the 10th ACM International Conference on Multimedia*, pages 570–579, Juan les Pins, France.

[Pohle, 2005] Pohle, T. (2005). Extraction of Audio Descriptors and Their Evaluation in Music Classification Tasks. Diplomarbeit, TU Kaiserslautern, DFKI, OFAI.

[Pohle, 2009] Pohle, T. (2009). *Automatic Characterization of Music for Intuitive Retrieval*. Dissertation, Johannes Kepler University, Linz, Austria.

[Pohle et al., 2007a] Pohle, T., Knees, P., Schedl, M., Pampalk, E., and Widmer, G. (2007a). "Reinventing the Wheel": A Novel Approach to Music Player Interfaces. *IEEE Transactions on Multimedia*, 9(3):567–575.

[Pohle et al., 2006] Pohle, T., Knees, P., Schedl, M., and Widmer, G. (2006). Automatically Adapting the Structure of Audio Similarity Spaces. In *Proceedings of 1st Workshop on Learning the Semantics of Audio Signals (LSAS'06)*, Athens, Greece.

[Pohle et al., 2007b] Pohle, T., Knees, P., Schedl, M., and Widmer, G. (2007b). Building an Interactive Next-Generation Artist Recommender Based on Automatically Derived High-Level Concepts. In *Proceedings of the 5th International Workshop on Content-Based Multimedia Indexing (CBMI'07)*, Bordeaux, France.

[Pohle and Schnitzer, 2007] Pohle, T. and Schnitzer, D. (2007). Striving for an Improved Audio Similarity Measure. In *4th Annual Music Information Retrieval Evaluation Exchange (MIREX'07)*.

[Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Rabiner and Juang, 1993] Rabiner, L. and Juang, B.-H. (1993). *Fundamentals of Speech Recognition*. Prentice-Hall, Upper Saddle River, NJ, USA.

[Rauber, 1999] Rauber, A. (1999). LabelSOM: On the Labeling of Self-Organizing Maps. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'99)*, Washington, DC, USA.

[Rauber and Frühwirth, 2001] Rauber, A. and Frühwirth, M. (2001). Automatically Analyzing and Organizing Music Archives. In *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries (ECDL'01)*, Darmstadt, Germany.

[Rauber et al., 2002] Rauber, A., Merkl, D., and Dittenbach, M. (2002). The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13:1331–1341.

[Rocchio, 1971] Rocchio, J. J. (1971). Relevance feedback in information retrieval. In Salton, G., editor, *The SMART Retrieval System – Experiments in Automatic Document Processing*. Prentice Hall Inc., Englewood Cliffs, NJ, USA.

[Rubner et al., 1998] Rubner, Y., Tomasi, C., and Guibas, L. (1998). A Metric for Distributions with Applications to Image Databases. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 59–66.

[Salton and Buckley, 1988] Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523.

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of 10th International Conference on World Wide Web (WWW'01)*, pages 285–295.

[Schedl, 2003] Schedl, M. (2003). An Explorative, Hierarchical User Interface to Structured Music Repositories. Master's thesis, Vienna University of Technology.

[Schedl, 2008] Schedl, M. (2008). *Automatically Extracting, Analyzing, and Visualizing Information on Music Artists from the World Wide Web*. PhD thesis, Johannes Kepler University Linz, Linz, Austria.

[Schedl, 2010] Schedl, M. (2010). On the Use of Microblogging Posts for Similarity Estimation and Artist Labeling. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, the Netherlands.

[Schedl and Knees, 2009] Schedl, M. and Knees, P. (2009). Context-based Music Similarity Estimation. In *Proceedings of the 3rd International Workshop on Learning the Semantics of Audio Signals (LSAS 2009)*, Graz, Austria.

[Schedl et al., 2006a] Schedl, M., Knees, P., Pohle, T., and Widmer, G. (2006a). Towards Automatic Retrieval of Album Covers. In *Proceedings of the 28th European Conference on Information Retrieval (ECIR'06)*, London, UK.

[Schedl et al., 2007] Schedl, M., Knees, P., Seyerlehner, K., and Pohle, T. (2007). The CoMIRVA Toolkit for Visualizing Music-Related Data. In *Proceedings of the 9th Eurographics/IEEE VGTC Symposium on Visualization (EuroVis'07)*, Norrköping, Sweden.

[Schedl et al., 2005a] Schedl, M., Knees, P., and Widmer, G. (2005a). A Web-Based Approach to Assessing Artist Similarity using Co-Occurrences. In *Proceedings of the 4th International Workshop on Content-Based Multimedia Indexing (CBMI'05)*, Riga, Latvia.

[Schedl et al., 2005b] Schedl, M., Knees, P., and Widmer, G. (2005b). Discovering and Visualizing Prototypical Artists by Web-based Co-Occurrence Analysis. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK.

[Schedl et al., 2005c] Schedl, M., Knees, P., and Widmer, G. (2005c). Improving Prototypical Artist Detection by Penalizing Exorbitant Popularity. In *Proceedings of the 3rd International Symposium on Computer Music Modeling and Retrieval (CMMR'05)*, Pisa, Italy.

[Schedl and Pohle, 2010] Schedl, M. and Pohle, T. (2010). Enlightening the Sun: A User Interface to Explore Music Artists via Multimedia Content. *Multimedia Tools and Applications*, 49(1).

[Schedl et al., 2006b] Schedl, M., Pohle, T., Knees, P., and Widmer, G. (2006b). Assigning and Visualizing Music Genres by Web-based Co-Occurrence Analysis. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR'06)*, Victoria, Canada.

[Schedl et al., 2010a] Schedl, M., Pohle, T., Koenigstein, N., and Knees, P. (2010a). What's Hot? Estimating Country-Specific Artist Popularity. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR'10)*, Utrecht, Netherlands.

[Schedl et al., 2010b] Schedl, M., Schiketanz, C., and Seyerlehner, K. (2010b). Country of Origin Determination via Web Mining Techniques. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'10): 2nd*

International Workshop on Advances in Music Information Research (AdMIRe 2010), Singapore.

[Schedl and Widmer, 2007] Schedl, M. and Widmer, G. (2007). Automatically Detecting Members and Instrumentation of Music Bands via Web Content Mining. In *Proceedings of the 5th Workshop on Adaptive Multimedia Retrieval (AMR'07)*, Paris, France.

[Schnitzer et al., 2010] Schnitzer, D., Flexer, A., Widmer, G., and Gasser, M. (2010). Islands of Gaussians: The Self Organizing Map and Gaussian Music Similarity Features. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR'10)*, Utrecht, Netherlands.

[Schnitzer et al., 2007] Schnitzer, D., Pohle, T., Knees, P., and Widmer, G. (2007). One-Touch Access to Music on Mobile Devices. In *Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia (MUM'07)*, pages 103–109, Oulu, Finland.

[Seyerlehner, 2006] Seyerlehner, K. (2006). Inhaltsbasierte Ähnlichkeitsmetriken zur Navigation in Musiksammlungen. Diplomarbeit, Johannes Kepler Universität, Linz, Austria.

[Seyerlehner et al., 2008] Seyerlehner, K., Widmer, G., and Knees, P. (2008). Frame level audio similarity - a codebook approach. In *Proceedings of the 11th International Conference on Digital Audio Effects (DAFx-08)*, Espoo, Finland.

[Shao et al., 2009] Shao, B., Wang, D., Li, T., and Ogihara, M. (2009). Music recommendation based on acoustic features and user access patterns. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(8):1602–1611.

[Shavitt and Weinsberg, 2009] Shavitt, Y. and Weinsberg, U. (2009). Songs Clustering Using Peer-to-Peer Co-occurrences. In *Proceedings of the IEEE International Symposium on Multimedia (ISM'09): International Workshop on Advances in Music Information Research (AdMIRe 2009)*, San Diego, CA, USA.

[Slaney and White, 2007] Slaney, M. and White, W. (2007). Similarity Based on Rating Data. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Smeaton et al., 2008] Smeaton, A. F., Wilkins, P., Worring, M., and de Rooij, O. (2008). Content-Based Video Retrieval: Three Example Systems from TRECVid. *International Journal of Imaging Science and Technology*, 18(2-3):195–201.

[Snoek et al., 2005] Snoek, C., Worring, M., and Smeulders, A. (2005). Early versus late fusion in semantic video analysis. In *Proceedings of the 13th ACM International Conference on Multimedia*, Singapore.

[Sordo et al., 2007] Sordo, M., Laurier, C., and Celma, O. (2007). Annotating Music Collections: How Content-based Similarity Helps to Propagate Labels. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, pages 531–534, Vienna, Austria.

[Stavness et al., 2005] Stavness, I., Gluck, J., Vilhan, L., and Fels, S. (2005). The MUSICtable: A Map-Based Ubiquitous System for Social Interaction with a Digital Music Collection. In *P*roceedings of the 4th International Conference on Entertainment Computing (ICEC 2005), Sanda, Japan.

[Stober and Nürnberger, 2010] Stober, S. and Nürnberger, A. (2010). MusicGalaxy - An Adaptive User-Interface for Exploratory Music Retrieval. In *Proceedings of 7th Sound and Music Computing Conference (SMC'10)*, pages 382–389, Barcelona, Spain.

[Tai, 1995] Tai, W. P. (1995). A Batch Training Network for Self-Organization. In Soulié, F. F. and Gallinari, P., editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN'95)*, volume II, pages 33–37, Nanterre, France. EC2.

[Torrens et al., 2004] Torrens, M., Hertzog, P., and Arcos, J.-L. (2004). Visualizing and Exploring Personal Music Libraries. In *Proceedings of 5th International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain.

[Tsymbalenko and Munson, 2001] Tsymbalenko, Y. and Munson, E. V. (2001). Using HTML Metadata to Find Relevant Images on the World Wide Web. *Internet Computing 2001*, pages 842–848.

[Turnbull et al., 2008a] Turnbull, D., Barrington, L., and Lanckriet, G. (2008a). Five Approaches to Collecting Tags for Music. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, Philadelphia, PA, USA.

[Turnbull et al., 2007a] Turnbull, D., Barrington, L., Torres, D., and Lanckriet, G. (2007a). Towards Musical Query-by-Semantic-Description using the CAL500 Data Set. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*, Amsterdam, Netherlands.

[Turnbull et al., 2008b] Turnbull, D., Barrington, L., Torres, D., and Lanckriet, G. (2008b). Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476.

[Turnbull et al., 2007b] Turnbull, D., Liu, R., Barrington, L., and Lanckriet, G. (2007b). A Game-based Approach for Collecting Semantic Annotations of Music. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07)*, Vienna, Austria.

[Typke, 2007] Typke, R. (2007). *Music Retrieval based on Melodic Similarity*. Rainer Typke.

[Tzanetakis and Cook, 2002] Tzanetakis, G. and Cook, P. (2002). Musical Genre Classification of Audio Signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302.

[van Gulik et al., 2004] van Gulik, R., Vignoli, F., and van de Wetering, H. (2004). Mapping music in the palm of your hand, explore and discover your collection. In *Proceedings of 5th International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain.

[Vapnik, 1995] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

[Vembu and Baumann, 2004] Vembu, S. and Baumann, S. (2004). A Self-Organizing Map Based Knowledge Discovery for Music Recommendation Systems. In *Proceedings of the 2nd International Symposium on Computer Music Modeling and Retrieval (CMMR'04)*, Esbjerg, Denmark.

[von Ahn and Dabbish, 2004] von Ahn, L. and Dabbish, L. (2004). Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'04)*, pages 319–326.

[Wang, 2006] Wang, A. (2006). The Shazam music recognition service. *Communications of the ACM*, 49(8):44–48.

[Whitman, 2005] Whitman, B. (2005). *Learning the meaning of music*. PhD thesis, Massachusetts Institute of Technology.

[Whitman and Lawrence, 2002] Whitman, B. and Lawrence, S. (2002). Inferring Descriptions and Similarity for Music from Community Metadata. In *Proceedings of the 2002 International Computer Music Conference (ICMC'02)*, pages 591–598, Gothenburg, Sweden.

[Whitman and Smaragdis, 2002] Whitman, B. and Smaragdis, P. (2002). Combining musical and cultural features for intelligent style detection. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR'02)*, pages 47–52, Paris, France.

[Wikipedia, 2010b] Wikipedia (2010b). "Information retrieval" – Wikipedia, The Free Encyclopedia.
URL: `http://en.wikipedia.org/w/index.php?title=Information_retrieval&oldid=357136854` [Online; accessed 03-May-2010].

[Wikipedia, 2010c] Wikipedia (2010c). "List of Music Genome Project attributes" — Wikipedia, The Free Encyclopedia.
URL: `http://en.wikipedia.org/w/index.php?title=List_of_Music_Genome_Project_attributes&oldid=365280477` [Online; accessed 14-June-2010].

[Wikipedia, 2010d] Wikipedia (2010d). "Music Genome Project" — Wikipedia, The Free Encyclopedia.
URL: `http://en.wikipedia.org/w/index.php?title=Music_Genome_Project&oldid=365713592` [Online; accessed 14-June-2010].

[Wikipedia, 2010e] Wikipedia (2010e). "Parsons code" – Wikipedia, The Free Encyclopedia.
URL: `http://en.wikipedia.org/w/index.php?title=Parsons_code&oldid=383375382` [Online; accessed 11-Oct-2010].

[Wikipedia, 2010f] Wikipedia (2010f). "Sound" – Wikipedia, The Free Encyclopedia.
URL: http://en.wikipedia.org/w/index.php?title=Sound&oldid=354265025
[Online; accessed 03-May-2010].

[Williams, 1983] Williams, L. (1983). Pyramidal parametrics. *SIGGRAPH Computer Graphics*, 17(3).

[Yang and Pedersen, 1997] Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, pages 412–420, Nashville, US.

[Yi et al., 2003] Yi, L., Liu, B., and Li, X. (2003). Eliminating noisy information in web pages for data mining. In *Proceedings of the 9th ACM SIGKDD Conference*.

[Zadel and Fujinaga, 2004] Zadel, M. and Fujinaga, I. (2004). Web Services for Music Information Retrieval. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR'04)*, Barcelona, Spain.

[Zhai and Liu, 2005] Zhai, Y. and Liu, B. (2005). Web data extraction based on partial tree alignment. In *Proceedings of the 14th International Conference on World Wide Web (WWW'05)*, pages 76–85, Chiba, Japan.

[Zobel and Moffat, 2006] Zobel, J. and Moffat, A. (2006). Inverted Files for Text Search Engines. *ACM Computing Surveys*, 38:1–56.

Appendix A

A.1 Supplementary Material

Music vocabulary consisting of 944 entries

17th century
18th century
19th century
20th century
2step
4 beat
50s
60s
70s
80s
90s
A Cappella
accidental
Accordion
acerbic
Acid
Acid House
Acid Jazz
Acid Rock
acoustical
adolescent
adult contemporary
aesthetic
affectionate
affective
African
African Jazz
Afro
Afro Cuban Jazz
aggressive
agitated
allegorical
alone
Alt Country
Alternative
Alternative Country
Alternative Dance
Alternative Folk
Alternative Metal
Alternative Rap
Alternative Rock
Ambient
Ambient Groove
Ambient Pop
american
Americana
American Punk
American Trad Rock
American Underground
AM Pop
amusing
Anarchist Punk
ancestral
ancient
andean
Dance Band
Dance Bands
Dancefloor
Dancehall
Dancehall reggae
dangerous
danish
Dark Metal
Darkstep
Dark Wave
Death Metal
Deep House
dejected
depressive
deranged
desolate
despairing
destructive
Detroit Rock
Detroit Techno
devotional
Didjeridu
Dirty Rap
Dirty South
Disco
disgusting
displeasing
dissatisfying
distraught
distrustful
disturbed
divine
Dixieland
Dixieland Jazz
Dixieland Revival
Doom Metal
Downbeat
Downtempo
dramatic
dreamlike
Dream Pop
dreamy
Drill and Bass
druggy
Drum
Drum and Bass
Drum n Bass
Dub
dutch
dynamic
dynamics
Early Creative
Early music
earthy
East Coast
imaginative
impolite
impressive
improvisation
improvisatory
Improvised Music
impulsive
indecent
Indie Pop
Indie Rock
indulgent
Industrial
Industrial Dance
Industrial Metal
injurious
Instrumental
Instrumental Country
Instrumental Rock
intellectual
intelligent
Intelligent Dance Music
intense
intimate
introspective
irish
ironic
israeli
italian
Italian Pop
Jam Bands
Jangle Pop
japanese
Japanese Pop
J Pop
Japanese Rock
Jazz
Jazz Funk
Jazz Pop
Jazz Rap
Jazz Rock
Jazzrock
Jingles
Jive
jovial
joyful
joyous
Jungle
Junkanoo
kaleidescopic
Klezmer
knotty
Kompa
korean
Kraut Rock
Kwaito
Punk Pop
Punk Revival
Punk Rock
Ragga
Ragtime
Ramshackle
Rap
Rapcore
Rap Metal
Rap Rock
Rave
raving
R&B
rebellious
Reed Organ
Reedpipe
Reggae
Reggaeton
relaxed
religious
Renaissance
repellent
repulsive
responsive
restless
restrained
Retro Rock
Retro Swing
rhapsodic
Rhythm and Blues
rhythmic
R n B
RnB
Rock
Rockabilly
Rockabilly Revival
Rock and Roll
Rock en espanol
Rock Hop
Rock & Roll
Rock Steady
Rocksteady
Rococco
romantic
Roots Rock
Rotterdam Techno
russian
sad
Sadcore
Salsa
Samba
sarcastic
Saxhorn
Saxophone
Saz

Continued on next page

Music vocabulary consisting of 944 entries (continued)

angry
angst ridden
anguished
animated
animistic
annoying
antisocial
anxious
archaic
archival
Arena Rock
art
artistic
Art Rock
art song
Asian Pop
atmospheric
Aussie Rock
austere
australian
autumnal
Avant garde Jazz
Background Music
bad mannered
Bagpipe
Bakersfield Sound
Ballad
Ballroom
Ballroom Dance
Bandoneon
Banjo
Baritone
baroque
Baroque Pop
Barrel Drum
Bass
Bass Drum
Bass Music
Beats
Bebop
belgian
Bhangra
biblical
Big Band
Big Beat
binary
Bitpop
Black Metal
bleak
blissful
bloody
Bluegrass
Blues
boisterous
Bongo
Boogie Rock
Boogie Woogie
Bop
Bossa Nova
bouncy
Bouzouki
Boy Band
Brachial Metal
brash
Brass Band
brassy
bravado
brazilian
Brazilian Jazz
Breakbeat
Breakbeat Hardcore
Breakcore
breezy
Brill Building Pop
brisk
british
British Blues
British Dance Bands
British Folk Rock
East Coast Rap
eastern
Easy Listening
effervescent
Electro
Electroclash
Electronic
Electronica
Electronic Body Music
Electronic Dance
Electropop
elegant
Emo
emotional
emphatic
endearing
energetic
enigmatic
Enka
entertaining
environmental
epic
erotic
eternal
ethereal
ethical
ethnic
euphoric
Eurodance
european
Euro Pop
Europop
Euro Rock
everchanging
evocative
excited
exciting
existential
exotic
experimental
Experimental Big Band
Experimental Rock
expressionistic
expressive
extensive
exuberant
fearful
fearsome
feminine
ferocious
Fiddle
fierce
fiery
filipino
Filk
finnish
Flageolet
Flamenco
Flamenco Guitar
Flute
Folk
Folk Rock
forceful
Foreign Language Pop
Foreign Language Rock
Foreign Rap
Fortepiano
fractured
frantic
Frat Rock
Freakbeat
freakish
Freeform
Freeform Hardcore
Free Funk
Free Jazz
Freestyle
Freestyle house
Freetekno
L.A. Punk
LA Punk
latin
Latin Continuum
Latin Jazz
Latin Pop
Latin Rap
Latin Rock
Lo Fi
loony
lovely
loving
Lutes
lyrical
macabre
Mainstream
Mainstream Jazz
Makina
Mandolin
manic
mannered
Mariachi
Marimba
masculine
materialistic
maternal
Math Rock
M Base
meandering
mechanical
mechanistic
Medieval
Meditation
meditative
melancholic
melancholy
mellow
Melodeon
Melodica
Melodic Metal
Melodic Trance
Melodiflute
Melodina
melodramatic
memorable
menacing
mendacious
merengue
Metal
mexican
Miami Bass
Microhouse
microtonal
middle aged
Middle Eastern Pop
minimalism
Minimalist Trance
miserable
Modal Music
Modern Big Band
Modern Creative
Modern Free
modernistic
Modern Jazz
Mod
modulations
moody
Mouth Organ
Musicals
Musique concrete
mystical
mythical
Nashville Sound
nasty
native american
Natural Trumpet
Neo Bop
Neo Classical Metal
Neo Glam
Scandinavian Metal
scary
Schlager
schmaltzy
science fictional
scientific
scurrilous
senegalese
sensual
sentimental
Serious Music
Serpahine
sexy
shakers
shamanistic
Shamisen
Shibuya kei
sinister
Sitar
Ska
Ska Punk
Ska Revival
Skatepunk
Skiffle
skittish
sleazy
slick
slovenian
Slowcore
Sludge Metal
slushy
Smooth Jazz
smouldering
Snare Drum
Soft Rock
somber
Songwriter
soothing
sophisticated
Sophisti Pop
Sopranino
Soprano
sorrowful
Soul
Soul Jazz
south african
Southern Gospel
Southern Rap
Southern Rock
Space Rock
spanish
spectacular
Speed Garage
Speed Metal
Spinet
spirited
Spiritual
Spirituals
spooky
Square Dance
Steam Organ
Steel Drum
stellar
Stoner Metal
Stoner Rock
strange
String
String Band
String Bands
stylish
Sunshine Pop
Surf Revival
Surf Rock
swedish
Swedish Pop
Swing
swiss
Symphonic Black Metal
Symphony

Continued on next page

Music vocabulary consisting of 944 entries (continued)

British Invasion
British Metal
British Psychedelia
British Pop
British Punk
British Rap
British Trad Rock
Britpop
Brit Pop
brittle
Brokenbeat
brooding
brotherly
Bubblegum
bustling
busy
byronic
Cabaret
Cajun
calm
Calypso
canadian
Candombe
caribbean
Castanets
casual
catastrophic
cathartic
Cathedral
celebratory
Cello
Celtic
Chamber Music
Chamber Pop
changeable
Chanson
chanting
Chapel
Chicago House
Chicano Rap
chill
Chillout
Chimes
chinese
Chinese Opera
Chinese Pop
Chinese Pop Rock
Chip Music
chirpy
Choral
Christian Rock
Christmas
Church
Clappers
Clarinet
Classical
Classical Guitar
Classic Jazz
Classic Soul
Claves
close harmony
College Rock
Comedy Rock
comical
compassionate
compelling
complex
Concertina
confident
confrontational
Congo
contemplative
Contemporary Music
Contemporary Bluegrass
Contemporary Country
contemptuous
Continental Jazz
Contrabass
contralto
freewheeling
french
French Pop
French Rock
frisky
Fugue
Functional Music
Funk
Funk Metal
funny
Fusion
Futurepop
futuristic
Gabba
Gangsta
Gangsta Rap
Garage
Garage Punk
Garage Rock
german
G Funk
ghastly
Ghetto
Ghetto House
Ghettotech
Girl Group
glamorous
Glam Rock
gleeful
Glitch
Glitter
Glockenspiel
gloomy
Goa
Goa Trance
Go Go
Golden Age
Gong
Gospel
Gospel Spiritual
Gothic
Gothic Metal
Goth Metal
Goth Rock
Grand Piano
greasy
greek
greenlandic
Gregorian
Grindcore
gritty
grotesque
gruesome
Grunge
Guitar
Guitar Virtuoso
gutsy
Gypsy
Hair Metal
happy
Happy Hardcore
Hard Bop
Hardcore
Hardcore Punk
Hardcore Rap
Hardcore Techno
Hard House
Hard Rock
Hardrock
Hard Stuff
Hard Trance
Harmonica
Harmony
Harp
Harpsichord
harsh
haunting
hazardous
Heartland Rock
Neo Prog
Neo Psychedelia
Neo Traditional Folk
Neo Traditionalist
Nerdcore
Neue Deutsche Welle
NDW
New Age
New Beat
New Orleans Brass
New Orleans Jazz
New Romantic
New School
New Traditionalist
New Wave
New York Punk
New Zealand Rock
nihilistic
nocturnal
noise
Noise Pop
Noise Rock
Nonet
norwegian
Novelty Ragtime
No Wave
Nu Jazz
Nu Metal
Nu NRG
Nu Soul
Oboe
Ocarina
Old School
Old School EBM
Old School Rap
Old Skool
Old Style
Old Timey
Opera
Operetta
Orchestra
Orchestral Jazz
Organ
oriental
Outlaw Country
outraged
outrageous
Panpipe
paranoid
Party Rap
pastoral
paternal
patriotic
Percussion
percussive
pessimistic
philosophical
Piano
Pimp
Pipe Organ
poetic
polish
political
Political Rap
Polka
Pop
Pop Metal
Pop Rap
Popular Music
Pop Underground
portuguese
Post Bop
Post Grunge
Post Horn
Post Punk
Post Rock
powerful
Power Metal
Power Pop
Synthesizer
Synth Pop
Synthpop
Synthpunk
Tabla
Tabor
taiwanese
Tambourine
Tambura
Tango
tanzanian
Tech House
Techno
Tenor
Thrash
Thrash Metal
thrilling
throbbing
thuggish
Timba
Timbale
tonal
traditional
Traditional Country
Trad Jazz
tragic
Trance
trancelike
trashy
Triangles
Trip Hop
trippy
Trombone
Truck Driving Country
Truck
True Metal
Trumpet
Trumpets
Tuba
Tubular Trumpet
turkish
Turntable
Twee Pop
Twelve String
twitchy
Tzouras
uncompromising
unconventional
uncreative
uncultured
Underground
Underground Rap
unethical
unexpressive
unfriendly
unfunny
unhappy
unhealthy
unimaginative
unintelligent
unintuitive
unnatural
unpleasant
unreal
unusual
Upbeat
uplifting
Urban
utopian
Vallenato
Variation
Variations
Vibraphone
vigorous
Viola
violent
Violin
vivacious
vocal

Continued on next page

Music vocabulary consisting of 944 entries (continued)

contrapuntal	Heavy Metal	primitive	vocalese
controversial	hedonistic	Prog	Vocal House
convulsive	hideous	Progressive Big Band	volatile
cool	hilarious	Progressive Bluegrass	vulgar
Country	Hi NRG	Progressive Country	Wave Metal
Country Boogie	HipHop	Progressive House	weird
Country Folk	Hip Hop	Progressive Jazz	welsh
Country Gospel	Hip House	Progressive Metal	West Coast
Country Music	Hong Kong Pop	Progressive Rock	West Coast Jazz
Countrypolitain	Honky Tonk	Progressive Trance	West Coast Rap
Country Pop	Horn	Prog Rock	Western Swing
Country Rock	Hornpipe	propulsive	Whistle
courageous	Horrorcore	Proto Punk	Wind Instrument
courteous	Hot Jazz	provocative	Wood Blocks
Cowbell	Hot Rod	Psychedelic	World Fusion
Cowpunk	House	Psychedelic Pop	worldwide
Crossover	humorous	Psychedelic Trance	Xylophone
Crossover Jazz	humourous	Psychobilly	yearning
cuban	hypnotic	Pub Rock	Yodeling
Cuban Jazz	icelandic	puerto rican	Yodelling
Cymbal	IDM	pulsating	youthful
cynical	idyllic	Punk	Zither
Dance	Illbient	Punk Metal	Zydeco

Table A.1: Music vocabulary consisting of 944 entries.

female vocalists (5056)	party (1763)	progressive (771)	thrash (236)
female vocalist (5056)	romantic (1737)	ska (763)	melodic death metal (215)
mellow (5003)	grunge (1737)	synthpop (748)	black metal (211)
singer-songwriter (4914)	60s (1694)	synth pop (748)	darkwave (205)
singer songwriter (4914)	piano (1664)	political (732)	avantgarde (197)
80s (4874)	vocal (1654)	atmospheric (723)	avant-garde (197)
hard rock (4826)	hardcore (1604)	metalcore (677)	breakbeat (194)
chill (4464)	female vocals (1581)	garage rock (675)	shoegaze (184)
00s (4096)	indie pop (1560)	alt-country (660)	dancehall (182)
pop-rock (4081)	new wave (1479)	christian rock (658)	dream pop (169)
pop rock (4081)	live (1424)	psychedelic rock (646)	french (148)
metal (3975)	usa (1392)	trance (634)	melodic metal (148)
hip hop (3898)	folk-rock (1377)	latin (633)	german (147)
hiphop (3898)	folk rock (1377)	dreamy (627)	drum and bass (147)
hip-hop (3898)	britpop (1376)	house (616)	doom metal (142)
punk (3894)	psychedelic (1375)	melodic (602)	remix (138)
soul (3867)	rhythm and blues (1327)	thrash metal (539)	swedish (136)
dance (3852)	alternative metal (1271)	gothic (538)	anime (134)
chillout (3720)	uk (1270)	progressive metal (526)	world music (124)
indie rock (3657)	nu metal (1260)	emocore (524)	electroclash (122)
classical (3587)	nu-metal (1260)	post-hardcore (520)	speed metal (117)
rnb (3289)	experimental (1236)	art rock (519)	idm (108)
relaxing (3154)	ambient (1217)	guitar virtuoso (470)	psytrance (107)
relax (3154)	melancholic (1215)	irish (463)	progressive trance (101)
country (3130)	pop punk (1211)	death metal (458)	noise (97)
british (3081)	rock n roll (1199)	swing (440)	bossa nova (95)
electronic (3035)	instrumental (1162)	world (436)	symphonic metal (86)
emo (2960)	blues rock (1104)	industrial metal (425)	contemporary classical (79)
70s (2894)	alternative punk (1085)	ethereal (421)	japanese (75)
guitar (2815)	funky (1053)	goth (373)	brazilian (74)
sad (2808)	dark (1034)	stoner rock (372)	dark electro (74)
oldies (2787)	post-punk (1001)	new age (342)	finnish (74)
cool (2781)	english (992)	rockabilly (342)	psychobilly (65)
folk (2775)	calm (986)	ska punk (341)	ebm (60)
acoustic (2724)	lounge (951)	power metal (339)	italian (57)
blues (2696)	canadian (928)	lo-fi (330)	grindcore (56)
rap (2680)	techno (900)	post-rock (325)	drum n bass (54)
punk rock (2629)	reggae (885)	post rock (325)	sludge (48)
sexy (2576)	industrial (872)	christmas (315)	brutal death metal (45)
happy (2333)	comedy (862)	electropop (307)	female fronted metal (44)
easy listening (2226)	glam rock (844)	spanish (300)	minimal (43)
jazz (2223)	disco (843)	gothic rock (291)	dnb (38)
ballad (2186)	epic (835)	dub (287)	deutsch (38)
soft rock (2113)	rock and roll (833)	gothic metal (278)	melodic black metal (31)
melancholy (2085)	downtempo (830)	industrial rock (277)	russian (30)
electronica (2022)	americana (823)	celtic (276)	jpop (29)
heavy metal (1876)	trip hop (818)	acid jazz (261)	jrock (29)
progressive rock (1856)	trip-hop (818)	hardcore punk (249)	j-rock (29)
catchy (1808)	screamo (815)	fusion (239)	j-pop (29)
funk (1781)	electro (806)	australian (236)	drone (24)

Table A.2: Evaluation queries for the c35k collection sorted according to the number of relevant tracks per query.

male lead vocals (339)	piano (84)	rap (21)
electric texture (326)	romantic (76)	string ensemble (20)
acoustic texture (278)	touching (75)	reading (20)
drum set (275)	pop (72)	breathy vocals (20)
high energy (231)	very danceable (68)	rapping (20)
comfortable (184)	distorted electric guitar (65)	intensely listening (20)
pleasant (184)	at a party (62)	romancing (19)
positive feelings (172)	aggressive (58)	horn section (18)
catchy (165)	sad (58)	contemporary r&b (16)
memorable (165)	acoustic guitar (58)	screaming (15)
bass (164)	going to sleep (56)	call and response (15)
synthesized texture (160)	electronica (56)	piano solo (14)
passionate (160)	soft rock (48)	at work (14)
strong (160)	angry (48)	cool jazz (13)
powerful (160)	drum machine (44)	hand drums (12)
emotional (160)	cleaning the house (43)	gravelly vocals (12)
awakening (154)	male lead vocals solo (40)	tambourine (12)
arousing (154)	sequencer (39)	falsetto (11)
backing vocals (153)	distorted electric guitar solo (36)	funk (11)
soothing (148)	r&b (36)	harmonica (10)
calming (148)	changing energy level (36)	saxophone solo (10)
driving (141)	altered with effects (35)	bluegrass (10)
rock (136)	high-pitched vocals (35)	spoken (10)
happy (135)	hanging with friends (34)	electric blues (9)
fast tempo (135)	studying (33)	violin (9)
heavy beat (130)	country (33)	brit pop (9)
electric guitar (124)	hard rock (32)	fiddle (9)
positive (120)	samples (32)	sleeping (8)
optimistic (120)	ambient sounds (32)	roots rock (8)
exciting (117)	metal (32)	waking up (8)
thrilling (117)	jazz (32)	trombone (8)
carefree (109)	folk (30)	alternative folk (8)
mellow (109)	soul (29)	gospel (7)
lighthearted (109)	getting ready to go out (29)	contemporary blues (7)
laid-back (109)	low-pitched vocals (27)	female lead vocals solo (7)
cheerful (107)	singer songwriter (25)	bebop (6)
festive (107)	blues (24)	monotone vocals (6)
soft (104)	punk (23)	organ (6)
tender (104)	saxophone (23)	duet (6)
alternative (100)	weird (22)	harmonica solo (6)
synthesizer (99)	exercising (22)	trumpet solo (6)
loving (98)	bizarre (22)	country blues (6)
emotional vocals (95)	hip hop (21)	acoustic guitar solo (6)
playful (92)	dance pop (21)	with the family (5)
light (92)	electric guitar solo (21)	swing (5)
female lead vocals (90)	world (21)	
classic rock (90)	trumpet (21)	

Table A.3: Evaluation queries for the CAL500 set sorted according to the number of relevant tracks per query.

"plastic band"	jazz
80ies synth pop	latin pop
ac/dc	mass in b minor
acdc	melodic metal with opera singer as front woman
american folk	metal
angry samoans	metallica
barbie girl	ndw
cello	neomedieval music
comedy	new orleans
dancehall	new zork scene
don't dream it's over	no new york
drude	nur die besten sterben jung
eurodance	oldies slow jazz
female electro	postmodern
filmmusik	punk
gangsta	punk rock
german hip hop	rammstein music with strong keyboard
ghost dog	rem
green day	schoenheitsfehler
groove	sicherheitsmann
guitar rock brit pop	soundtrack
happy sound	vienna electro dj
hard rock fast guns'n roses	violin
heavy metal with orchestra	weilheim
herr lehmann	wie lieblich sind deine wohnungen
in extremo live	world
indie rock	zztop
industrial rock trent reznor	

Table A.4: 55 queries issued by users in the user study.

A.2 Detailed Evaluation Results

Explanation for all tables: All entries are obtained by calculating the mean result over all queries. Values are given in percent. A bold typeface indicates that the entry belongs to the group of significantly best settings (per line) according to the Friedman test with a significance level $\alpha = 0.01$. The structure of this appendix section corresponds to the structure of Section 5.6.2.

A.2.1 Web Search Engine Impact

	Rec		Prec		Prec@10	
Baseline	100.00		3.65		3.60	
	Google	exalead	Google	exalead	Google	exalead
$RRS_{n=10}$	2.18	1.73	30.15	28.42	31.19	27.40
$RRS_{n=20}$	3.74	2.80	29.02	28.05	32.40	30.30
$RRS_{n=50}$	7.17	5.46	27.61	26.43	38.45	32.35
$RRS_{n=100}$	12.72	8.71	25.99	25.06	44.10	35.90
$RRS_{n=200}$	18.67	13.40	23.77	23.01	47.75	33.80
$RRS_{n=500}$	29.31	23.19	20.12	20.41	50.30	34.60
$RRS_{n=1000}$	40.38	34.01	16.88	17.97	52.55	36.80
$RRS_{n=10000}$	80.50	72.48	7.29	8.81	57.45	42.55
$PseudoDoc$	93.66	87.20	4.27	5.52	39.25	34.95
	rPrec		AvgPrec		AUC	
Baseline	3.65		3.68		4.18	
	Google	exalead	Google	exalead	Google	exalead
$RRS_{n=10}$	2.16	1.42	1.19	0.93	3.05	2.79
$RRS_{n=20}$	3.63	2.48	1.84	1.38	3.64	3.13
$RRS_{n=50}$	6.52	4.73	3.24	2.42	4.99	4.13
$RRS_{n=100}$	10.24	7.41	5.54	3.72	7.11	5.17
$RRS_{n=200}$	14.22	10.76	8.23	5.34	9.62	6.62
$RRS_{n=500}$	19.84	15.70	12.39	8.42	13.76	9.79
$RRS_{n=1000}$	24.22	20.36	16.10	11.96	17.22	13.36
$RRS_{n=10000}$	35.20	30.25	29.98	23.97	31.25	25.32
$PseudoDoc$	30.78	26.72	25.97	21.16	27.09	22.29

Table A.5: Google vs. exalead Web search engine impact evaluated on c35k

	Rec		Prec		Prec@10	
Baseline	100.00		13.32		13.33	
	Google	exalead	Google	exalead	Google	exalead
$RRS_{n=10}$	5.96	5.58	25.77	26.74	25.77	26.80
$RRS_{n=20}$	10.19	9.13	24.87	24.11	25.98	25.42
$RRS_{n=50}$	17.99	15.68	22.84	22.11	26.06	26.55
$RRS_{n=100}$	26.80	23.66	21.02	20.63	29.30	27.99
$RRS_{n=200}$	38.63	33.42	19.15	18.72	30.60	28.63
$RRS_{n=500}$	56.31	47.36	16.86	17.20	32.68	29.28
$RRS_{n=1000}$	66.91	56.09	15.54	16.25	33.47	29.50
$RRS_{n=10000}$	73.27	61.43	14.56	15.45	33.62	30.43
$PseudoDoc$	81.15	70.52	14.50	15.08	30.72	26.47
	rPrec		AvgPrec		AUC	
Baseline	13.31		14.31		15.69	
	Google	exalead	Google	exalead	Google	exalead
$RRS_{n=10}$	5.61	5.33	3.58	2.78	4.50	3.77
$RRS_{n=20}$	8.84	7.65	5.30	4.16	6.13	5.00
$RRS_{n=50}$	13.49	11.88	7.57	6.34	8.63	7.22
$RRS_{n=100}$	18.05	15.81	10.59	8.52	11.69	9.61
$RRS_{n=200}$	21.58	18.77	13.84	10.85	15.16	12.13
$RRS_{n=500}$	24.06	20.77	18.02	14.15	19.33	15.51
$RRS_{n=1000}$	24.86	21.02	20.37	15.93	21.77	17.35
$RRS_{n=10000}$	25.06	21.12	21.77	16.96	23.16	18.27
$PseudoDoc$	25.77	21.31	22.66	17.57	24.12	19.21

Table A.6: Google vs. exalead Web search engine impact evaluated on CAL500

A.2.2 Page Filtering Impact

artist threshold	12	15	18	20	12	15	18	20
	Rec				**Prec**			
$RRS_{n=10}$	**2.10**	**2.07**	**2.13**	**2.16**	**33.50**	**35.72**	**35.09**	**34.03**
$RRS_{n=20}$	**3.88**	**3.93**	**3.70**	**3.66**	**32.91**	**32.86**	**32.45**	**32.23**
$RRS_{n=50}$	**8.09**	**7.87**	**7.74**	**7.86**	**31.26**	**31.13**	**30.91**	**30.64**
$RRS_{n=100}$	**12.14**	**12.58**	**12.39**	**12.58**	**28.61**	**29.05**	**28.62**	**28.40**
$RRS_{n=200}$	**18.29**	**18.13**	**18.09**	**18.01**	**26.19**	**26.21**	**25.86**	**25.61**
$RRS_{n=500}$	**29.22**	**29.84**	**29.29**	**29.25**	**21.44**	**21.71**	**21.65**	**21.63**
$RRS_{n=1000}$	**39.85**	**40.11**	**40.02**	**40.11**	**18.05**	**18.19**	**18.12**	**18.09**
$RRS_{n=10000}$	75.29	**76.80**	**77.55**	**78.01**	**8.00**	**7.87**	**7.85**	7.82
	Prec@10				**rPrec**			
$RRS_{n=10}$	**36.15**	**37.76**	**36.39**	**35.26**	**2.09**	**2.05**	**2.11**	**2.14**
$RRS_{n=20}$	**36.14**	**37.05**	**35.95**	**35.10**	**3.80**	**3.67**	**3.64**	**3.59**
$RRS_{n=50}$	**41.80**	**41.70**	**39.95**	**40.70**	**7.23**	**7.08**	**7.15**	**7.08**
$RRS_{n=100}$	**45.85**	**46.65**	**44.90**	**45.00**	**10.54**	**10.78**	**10.55**	**10.77**
$RRS_{n=200}$	**46.90**	**48.90**	**48.35**	**48.15**	**14.51**	**14.68**	**14.53**	**14.62**
$RRS_{n=500}$	**51.35**	**51.20**	**50.30**	**49.95**	**20.25**	**20.35**	**20.28**	**20.39**
$RRS_{n=1000}$	**52.35**	**52.85**	**52.90**	**52.60**	**25.08**	**25.00**	**24.72**	**24.70**
$RRS_{n=10000}$	**55.95**	**56.70**	**56.95**	**57.00**	34.64	**35.03**	**35.26**	**35.46**
	AvgPrec				**AUC**			
$RRS_{n=10}$	**1.36**	**1.38**	**1.37**	**1.33**	**3.28**	**3.34**	**3.32**	**3.27**
$RRS_{n=20}$	**2.24**	**2.24**	**2.16**	2.08	**4.01**	**4.07**	3.98	3.88
$RRS_{n=50}$	**4.10**	**4.06**	**4.00**	3.98	**5.96**	**5.86**	**5.89**	**5.86**
$RRS_{n=100}$	**6.15**	**6.29**	**6.13**	**6.23**	**7.79**	**7.91**	**7.76**	**7.91**
$RRS_{n=200}$	**8.66**	**8.78**	**8.69**	**8.74**	**10.15**	**10.19**	**10.14**	**10.10**
$RRS_{n=500}$	**13.04**	**13.19**	**13.10**	**13.11**	**14.34**	**14.57**	**14.43**	**14.45**
$RRS_{n=1000}$	**16.95**	**17.01**	**16.85**	**16.85**	**18.26**	**18.36**	**18.15**	**18.18**
$RRS_{n=10000}$	29.33	29.80	**30.06**	**30.24**	30.47	**31.07**	**31.26**	**31.45**

Table A.7: Impact of 2MA artist occurrence threshold (Google, c35k)

artist threshold	5	10	15	20	5	10	15	20
	Rec				**Prec**			
$RRS_{n=10}$	**1.45**	**1.81**	**1.90**	**1.86**	**24.69**	**26.49**	**28.00**	**27.45**
$RRS_{n=20}$	**2.64**	**2.91**	**3.05**	**2.87**	**24.31**	**26.37**	**27.48**	**26.86**
$RRS_{n=50}$	**5.12**	**5.39**	**5.56**	**5.64**	**23.53**	**25.86**	**26.65**	**26.50**
$RRS_{n=100}$	**8.61**	**8.53**	**8.70**	**8.71**	**23.39**	**24.37**	**25.17**	**25.31**
$RRS_{n=200}$	**13.11**	**13.30**	**13.23**	**13.40**	22.01	**22.86**	**23.24**	**23.44**
$RRS_{n=500}$	**22.13**	**23.29**	**23.56**	**23.46**	**19.62**	**20.36**	**20.81**	**20.87**
$RRS_{n=1000}$	**29.33**	**33.09**	**33.92**	**33.82**	17.10	**18.11**	**18.39**	**18.49**
$RRS_{n=10000}$	49.71	63.80	**67.67**	**69.30**	**11.77**	10.36	9.77	9.60
	Prec@10				**rPrec**			
$RRS_{n=10}$	**26.07**	**27.59**	**26.66**	**26.69**	**1.45**	**1.79**	**1.58**	**1.54**
$RRS_{n=20}$	**27.33**	**31.59**	**29.19**	**28.50**	**2.55**	**2.87**	**2.62**	**2.53**
$RRS_{n=50}$	29.20	**31.44**	**33.27**	**32.55**	**4.73**	**5.18**	**4.88**	**4.97**
$RRS_{n=100}$	**29.60**	**33.00**	**33.25**	**33.95**	**7.33**	**7.49**	**7.73**	**7.65**
$RRS_{n=200}$	**32.05**	**33.60**	**32.90**	**32.25**	**10.38**	**10.78**	**10.49**	**10.68**
$RRS_{n=500}$	**33.50**	**34.95**	**35.60**	**34.15**	**15.51**	**16.34**	**16.38**	**16.42**
$RRS_{n=1000}$	**34.90**	**36.10**	**36.85**	**36.95**	18.63	**20.32**	**20.92**	**20.79**
$RRS_{n=10000}$	**36.05**	**36.95**	**39.40**	**39.40**	21.32	26.68	**28.69**	**29.42**
	AvgPrec				**AUC**			
$RRS_{n=10}$	**0.78**	**1.04**	**0.96**	**0.97**	**2.54**	**2.83**	**2.67**	**2.67**
$RRS_{n=20}$	**1.21**	**1.49**	**1.44**	**1.38**	**2.84**	**3.24**	**3.43**	**3.24**
$RRS_{n=50}$	**2.11**	**2.53**	**2.49**	**2.50**	**3.68**	**4.13**	**4.23**	**4.17**
$RRS_{n=100}$	**3.23**	**3.65**	**3.79**	**3.88**	**4.65**	**4.96**	**5.34**	**5.33**
$RRS_{n=200}$	**4.80**	**5.29**	**5.17**	**5.36**	**6.21**	**6.66**	**6.59**	**6.82**
$RRS_{n=500}$	**7.68**	**8.83**	**8.82**	**8.66**	**9.04**	**10.26**	**10.32**	**10.19**
$RRS_{n=1000}$	9.91	**11.99**	**12.37**	**12.21**	11.27	**13.31**	**13.79**	**13.71**
$RRS_{n=10000}$	13.64	19.81	**22.05**	**22.80**	14.96	21.15	**23.57**	**24.25**

Table A.8: Impact of 2MA artist occurrence threshold (exalead, c35k)

artist threshold	∞	5	10	12	15	18	20	30
	Rec							
$RRS_{n=10}$	**5.96**	**5.61**	**5.60**	**5.57**	**5.85**	**5.83**	**5.93**	**5.83**
$RRS_{n=20}$	**10.19**	8.92	**9.71**	**9.48**	**9.46**	**9.75**	**9.87**	**10.07**
$RRS_{n=50}$	**17.99**	15.17	16.61	**17.40**	**17.20**	**17.36**	**17.50**	**17.57**
$RRS_{n=100}$	**26.80**	23.28	24.85	25.29	**25.48**	**25.85**	**25.81**	**26.30**
$RRS_{n=200}$	**38.63**	30.81	36.47	36.91	37.24	**37.91**	**38.00**	**38.07**
$RRS_{n=500}$	**56.31**	40.97	51.34	53.17	54.26	54.69	**55.18**	**55.66**
$RRS_{n=1000}$	**66.91**	45.57	59.08	61.51	63.78	**64.99**	**65.55**	**66.15**
$RRS_{n=10000}$	**73.27**	47.10	63.16	66.08	69.06	70.43	71.11	**72.36**
	Prec							
$RRS_{n=10}$	**25.77**	**26.80**	**25.92**	**25.23**	**25.72**	**25.75**	**26.04**	**25.66**
$RRS_{n=20}$	**24.87**	**24.60**	**24.58**	**24.32**	**24.69**	**24.95**	**24.95**	**24.98**
$RRS_{n=50}$	**22.84**	**22.10**	**22.51**	**22.47**	**22.44**	**22.64**	**22.68**	**22.76**
$RRS_{n=100}$	**21.02**	**20.71**	**20.90**	**20.69**	**20.80**	**20.96**	**20.92**	**21.08**
$RRS_{n=200}$	**19.15**	**19.28**	**19.30**	**19.19**	**19.11**	**19.24**	**19.21**	**19.19**
$RRS_{n=500}$	**16.86**	**17.75**	**17.40**	**17.12**	**16.92**	**16.88**	**16.88**	**16.90**
$RRS_{n=1000}$	15.54	**17.03**	**16.29**	**15.91**	**15.74**	**15.63**	**15.60**	**15.59**
$RRS_{n=10000}$	14.56	**16.58**	**15.51**	**15.09**	14.82	14.69	14.64	14.61
	Prec@10							
$RRS_{n=10}$	**25.77**	**26.76**	**25.96**	**25.20**	**25.63**	**25.69**	**25.99**	**25.70**
$RRS_{n=20}$	**25.98**	**26.12**	**25.49**	**25.46**	**25.51**	**25.51**	**25.87**	**25.94**
$RRS_{n=50}$	**26.06**	**26.98**	**26.28**	**26.00**	**26.14**	**26.28**	**26.57**	**25.92**
$RRS_{n=100}$	**29.30**	**28.57**	**29.45**	**28.94**	**29.23**	**29.30**	**29.01**	**29.30**
$RRS_{n=200}$	**30.60**	**29.21**	**30.89**	**29.74**	**29.88**	**30.02**	**29.80**	**30.09**
$RRS_{n=500}$	**32.68**	29.43	**32.11**	**31.32**	**31.68**	**31.39**	**31.17**	**32.32**
$RRS_{n=1000}$	**33.47**	29.93	**32.69**	**32.54**	**32.61**	**32.83**	**32.68**	**33.40**
$RRS_{n=10000}$	**33.62**	29.93	**32.69**	**32.61**	**33.33**	**32.83**	**32.90**	**33.19**
	rPrec							
$RRS_{n=10}$	**5.61**	**5.32**	**5.51**	**5.21**	**5.50**	**5.47**	**5.58**	**5.48**
$RRS_{n=20}$	**8.84**	**8.10**	**8.42**	**8.45**	**8.54**	**8.52**	**8.64**	**8.72**
$RRS_{n=50}$	**13.49**	12.24	**12.97**	**12.95**	**13.08**	**13.17**	**13.26**	**13.42**
$RRS_{n=100}$	**18.05**	15.62	**17.51**	**17.86**	**17.81**	**17.65**	**17.80**	**17.90**
$RRS_{n=200}$	**21.58**	18.09	20.80	**21.10**	**21.14**	**21.28**	**21.26**	**21.34**
$RRS_{n=500}$	**24.06**	19.19	**23.11**	**23.22**	**23.59**	**23.76**	**23.73**	**24.06**
$RRS_{n=1000}$	**24.86**	19.48	23.55	**24.19**	**24.54**	**24.95**	**24.89**	**24.80**
$RRS_{n=10000}$	**25.06**	19.57	23.89	**24.49**	**24.96**	**24.87**	**24.96**	**25.08**
	AvgPrec							
$RRS_{n=10}$	**3.58**	**3.21**	**3.30**	**3.27**	**3.41**	**3.52**	**3.54**	**3.54**
$RRS_{n=20}$	**5.30**	**4.61**	**4.84**	**4.92**	**5.11**	**5.27**	**5.29**	**5.33**
$RRS_{n=50}$	**7.57**	7.04	**7.35**	**7.45**	**7.34**	**7.42**	**7.56**	**7.56**
$RRS_{n=100}$	**10.59**	9.17	**10.01**	**10.15**	**10.18**	**10.23**	**10.30**	**10.47**
$RRS_{n=200}$	**13.84**	11.29	**12.97**	**13.19**	**13.24**	**13.45**	**13.50**	**13.63**
$RRS_{n=500}$	**18.02**	13.56	**16.49**	**16.94**	**17.32**	**17.50**	**17.55**	**17.83**
$RRS_{n=1000}$	**20.37**	14.48	18.26	18.85	**19.48**	**19.76**	**19.91**	**20.22**
$RRS_{n=10000}$	**21.77**	14.69	19.19	20.03	20.75	**21.05**	**21.22**	**21.55**

Table A.9: Impact of 2MA artist occurrence threshold (Google, CAL500)

artist threshold	∞	5	10	12	15	18	20	30
	Rec							
$RRS_{n=10}$	**5.58**	4.92	**5.04**	**5.46**	**5.42**	**5.42**	**5.71**	**5.75**
$RRS_{n=20}$	**9.13**	**8.00**	**8.28**	**8.42**	**8.39**	**8.38**	**8.68**	**8.97**
$RRS_{n=50}$	**15.68**	14.65	**15.19**	**15.05**	**15.21**	**15.18**	**15.24**	**15.58**
$RRS_{n=100}$	**23.66**	20.24	22.55	22.92	22.86	**23.00**	**23.15**	**23.40**
$RRS_{n=200}$	**33.42**	25.78	30.84	31.69	32.34	**32.39**	**32.66**	**33.12**
$RRS_{n=500}$	**47.36**	33.85	42.66	44.16	45.40	45.79	46.07	**46.79**
$RRS_{n=1000}$	**56.09**	38.08	49.35	51.39	53.39	54.12	54.55	**55.52**
$RRS_{n=10000}$	**61.43**	40.24	52.94	55.69	58.17	59.15	59.66	**60.93**
	Prec							
$RRS_{n=10}$	**26.74**	**24.82**	**25.83**	**26.50**	**26.73**	**26.86**	**27.11**	**27.30**
$RRS_{n=20}$	**24.11**	**24.30**	**24.47**	**24.59**	**24.30**	**24.40**	**24.37**	**24.48**
$RRS_{n=50}$	**22.11**	**22.28**	**22.58**	**22.17**	**22.23**	**22.21**	**22.24**	**22.29**
$RRS_{n=100}$	**20.63**	**20.77**	**20.92**	**20.75**	**20.73**	**20.83**	**20.88**	**20.77**
$RRS_{n=200}$	18.72	**19.21**	**19.37**	**19.21**	**19.16**	**19.12**	**19.09**	**18.83**
$RRS_{n=500}$	17.20	**17.85**	**17.71**	**17.67**	**17.62**	**17.57**	**17.48**	**17.22**
$RRS_{n=1000}$	16.25	**17.27**	**16.75**	**16.58**	**16.53**	**16.45**	**16.39**	**16.19**
$RRS_{n=10000}$	15.45	**16.95**	**16.11**	**15.93**	**15.82**	15.69	15.62	15.36
	Prec@10							
$RRS_{n=10}$	**26.80**	**24.65**	**25.80**	**26.54**	**26.62**	**26.74**	**27.01**	**27.15**
$RRS_{n=20}$	**25.42**	**24.49**	**25.30**	**25.37**	**25.15**	**25.17**	**25.49**	**25.91**
$RRS_{n=50}$	**26.55**	**26.14**	**25.96**	**25.99**	**26.21**	**26.07**	**26.57**	**26.64**
$RRS_{n=100}$	**27.99**	**26.25**	**26.68**	**27.11**	**26.57**	**26.79**	**27.51**	**27.72**
$RRS_{n=200}$	**28.63**	**25.68**	**27.11**	**27.39**	**28.23**	**27.79**	**28.08**	**28.73**
$RRS_{n=500}$	**29.28**	25.89	**28.48**	**28.11**	**28.80**	**28.73**	**29.23**	**29.45**
$RRS_{n=1000}$	**29.50**	25.53	**28.62**	**28.90**	**29.88**	**29.38**	**29.59**	**29.74**
$RRS_{n=10000}$	**30.43**	26.32	**29.27**	**29.41**	**30.17**	**29.59**	**30.02**	**30.46**
	rPrec							
$RRS_{n=10}$	**5.33**	4.68	**5.04**	**5.32**	**5.26**	**5.35**	**5.35**	**5.39**
$RRS_{n=20}$	**7.65**	**7.00**	**7.23**	**7.57**	**7.57**	**7.31**	**7.39**	**7.49**
$RRS_{n=50}$	**11.88**	10.83	**11.36**	**11.45**	**11.63**	**11.65**	**11.50**	**11.94**
$RRS_{n=100}$	**15.81**	13.45	15.11	15.14	15.46	**15.49**	**15.71**	**15.73**
$RRS_{n=200}$	**18.77**	15.14	**17.67**	**17.69**	**18.31**	**18.33**	**18.50**	**18.78**
$RRS_{n=500}$	**20.77**	15.87	18.83	19.42	**20.09**	**20.36**	**20.43**	**20.63**
$RRS_{n=1000}$	**21.02**	16.07	19.24	**20.06**	**20.79**	**20.75**	**20.79**	**21.02**
$RRS_{n=10000}$	**21.12**	16.00	19.17	**20.01**	**20.53**	**20.58**	**20.78**	**21.04**
	AvgPrec							
$RRS_{n=10}$	**2.78**	**2.69**	**2.87**	**2.88**	**2.92**	**2.87**	**2.88**	**2.85**
$RRS_{n=20}$	**4.16**	**3.69**	**3.94**	**4.17**	**4.16**	**4.16**	**4.19**	**4.20**
$RRS_{n=50}$	**6.34**	**6.05**	**5.92**	**5.97**	**6.12**	**6.12**	**6.17**	**6.34**
$RRS_{n=100}$	**8.52**	7.57	**8.25**	**8.23**	**8.24**	**8.24**	**8.29**	**8.50**
$RRS_{n=200}$	**10.85**	8.93	**10.36**	**10.37**	**10.55**	**10.56**	**10.61**	**10.83**
$RRS_{n=500}$	**14.15**	10.42	12.94	13.26	**13.62**	**13.78**	**13.82**	**14.09**
$RRS_{n=1000}$	**15.93**	11.00	14.07	14.58	15.12	**15.32**	**15.41**	**15.81**
$RRS_{n=10000}$	**16.96**	11.24	14.82	15.40	15.94	16.19	**16.32**	**16.81**

Table A.10: Impact of 2MA artist occurrence threshold (exalead, CAL500)

artist threshold	∞	5	10	12	15	18	20	30
	Rec							
$RRS_{n=10}$	**6.10**	5.46	**6.04**	**5.90**	**6.01**	**6.10**	**6.08**	**6.11**
$RRS_{n=20}$	**9.68**	8.92	**9.54**	**9.59**	**9.48**	**9.61**	**9.71**	**9.62**
$RRS_{n=50}$	**18.09**	15.94	**17.22**	**17.46**	**17.59**	**17.57**	**17.65**	**17.88**
$RRS_{n=100}$	**26.34**	23.53	25.33	**25.59**	**25.91**	**26.26**	**26.19**	**26.28**
$RRS_{n=200}$	**38.62**	30.98	36.63	**37.20**	**37.27**	**37.80**	**37.94**	**38.26**
$RRS_{n=500}$	**55.65**	39.65	50.93	52.69	53.46	**54.42**	**54.71**	**55.05**
$RRS_{n=1000}$	**66.48**	43.47	58.16	60.54	62.87	64.07	**64.65**	**65.71**
$RRS_{n=10000}$	**72.93**	45.01	61.94	65.12	68.38	69.88	70.60	**72.02**
	Prec							
$RRS_{n=10}$	**27.17**	**25.66**	**27.12**	**26.86**	**27.22**	**27.31**	**27.32**	**27.42**
$RRS_{n=20}$	**25.39**	**24.42**	**25.19**	**25.22**	**25.36**	**25.38**	**25.45**	**25.45**
$RRS_{n=50}$	**23.14**	**22.45**	**22.94**	**22.88**	**22.87**	**22.84**	**22.84**	**23.07**
$RRS_{n=100}$	**21.05**	**21.01**	**21.17**	**21.10**	**21.10**	**21.15**	**21.06**	**21.14**
$RRS_{n=200}$	**19.30**	**19.24**	**19.24**	**19.27**	**19.19**	**19.22**	**19.23**	**19.29**
$RRS_{n=500}$	**16.95**	**17.92**	**17.40**	**17.20**	**17.00**	**16.95**	**16.93**	**16.96**
$RRS_{n=1000}$	15.81	**17.32**	**16.47**	**16.13**	**15.96**	**15.89**	**15.85**	**15.87**
$RRS_{n=10000}$	14.84	**16.92**	**15.79**	**15.38**	15.13	14.99	14.94	14.91
	Prec@10							
$RRS_{n=10}$	**27.17**	**25.66**	**27.07**	**26.80**	**27.16**	**27.32**	**27.32**	**27.42**
$RRS_{n=20}$	**26.38**	**25.05**	**26.20**	**26.32**	**26.53**	**26.67**	**26.74**	**26.60**
$RRS_{n=50}$	**27.00**	**25.89**	**26.73**	**26.86**	**26.71**	**27.00**	**27.21**	**27.14**
$RRS_{n=100}$	**30.16**	**30.07**	**30.13**	**29.52**	**29.95**	**30.24**	**30.24**	**29.95**
$RRS_{n=200}$	**31.53**	**30.86**	**31.33**	**31.46**	**30.38**	**30.81**	**31.17**	**31.24**
$RRS_{n=500}$	**32.32**	**31.15**	**32.49**	**32.61**	**31.68**	**31.96**	**32.18**	**32.47**
$RRS_{n=1000}$	**32.75**	**30.50**	**32.63**	**32.76**	**31.97**	**31.96**	**32.25**	**32.61**
$RRS_{n=10000}$	**33.11**	30.64	**33.06**	**33.62**	**32.61**	**32.25**	**32.61**	**32.83**
	rPrec							
$RRS_{n=10}$	**6.00**	5.46	**6.04**	**5.90**	**5.90**	**6.00**	**5.98**	**6.01**
$RRS_{n=20}$	**8.68**	7.75	**8.66**	**8.64**	**8.55**	**8.67**	**8.78**	**8.70**
$RRS_{n=50}$	**13.77**	12.54	**13.36**	**13.37**	**13.35**	**13.56**	**13.62**	**13.61**
$RRS_{n=100}$	**17.83**	15.99	**17.43**	**17.57**	**17.43**	**17.65**	**17.66**	**17.66**
$RRS_{n=200}$	**22.05**	17.89	**21.30**	**21.74**	**21.78**	**21.73**	**21.62**	**21.78**
$RRS_{n=500}$	**24.69**	19.10	**23.68**	**23.66**	**24.01**	**24.15**	**24.15**	**24.48**
$RRS_{n=1000}$	**25.43**	19.58	**24.45**	**24.74**	**24.69**	**25.08**	**25.09**	**25.27**
$RRS_{n=10000}$	**25.60**	19.60	24.53	**24.99**	**24.85**	**25.18**	**25.27**	**25.60**
	AvgPrec							
$RRS_{n=10}$	**3.46**	**3.12**	**3.37**	**3.37**	**3.44**	**3.47**	**3.48**	**3.47**
$RRS_{n=20}$	**4.88**	4.41	**4.78**	**4.81**	**4.82**	**4.85**	**4.88**	**4.88**
$RRS_{n=50}$	**8.03**	7.09	**7.60**	**7.72**	**7.82**	**7.91**	**7.94**	**7.99**
$RRS_{n=100}$	**10.96**	9.43	**10.26**	**10.54**	**10.65**	**10.86**	**10.86**	**10.93**
$RRS_{n=200}$	**14.02**	11.45	**13.20**	**13.59**	**13.59**	**13.87**	**13.88**	**13.98**
$RRS_{n=500}$	**18.20**	13.62	16.61	**17.18**	**17.36**	**17.67**	**17.77**	**18.04**
$RRS_{n=1000}$	**20.42**	14.43	18.31	19.05	19.35	**19.68**	**19.84**	**20.19**
$RRS_{n=10000}$	**22.04**	14.80	19.34	20.26	20.74	21.17	**21.37**	**21.83**

Table A.11: Impact of 2MA artist occurrence threshold on ANR (Google, CAL500)

artist threshold	∞	5	10	12	15	18	20	30
	Rec							
$RRS_{n=10}$	**4.65**	**4.59**	**4.48**	**4.61**	**4.62**	**4.61**	**4.65**	**4.62**
$RRS_{n=20}$	**8.46**	7.43	**7.98**	**8.24**	**8.24**	**8.26**	**8.29**	**8.35**
$RRS_{n=50}$	**15.63**	13.82	14.88	14.92	**15.10**	**15.07**	**15.15**	**15.52**
$RRS_{n=100}$	**23.92**	19.98	22.50	22.87	**23.25**	**23.32**	**23.42**	**23.83**
$RRS_{n=200}$	**34.26**	25.89	31.15	32.17	32.88	**33.25**	**33.32**	**34.08**
$RRS_{n=500}$	**47.22**	32.91	42.11	43.67	45.39	**45.86**	**46.06**	**47.05**
$RRS_{n=1000}$	**55.68**	36.49	48.67	50.74	52.67	53.62	**54.02**	**55.25**
$RRS_{n=10000}$	**60.28**	37.88	51.36	54.08	56.71	57.77	58.29	**59.74**
	Prec							
$RRS_{n=10}$	**24.40**	**24.87**	**24.02**	**24.20**	**24.18**	**24.21**	**24.42**	**24.48**
$RRS_{n=20}$	**23.97**	**23.02**	**23.61**	**23.90**	**23.74**	**23.89**	**23.98**	**23.92**
$RRS_{n=50}$	**21.87**	**21.50**	**21.60**	**21.60**	**21.69**	**21.65**	**21.75**	**21.81**
$RRS_{n=100}$	**20.77**	20.52	**20.92**	**20.80**	**20.89**	**20.78**	**20.76**	**20.70**
$RRS_{n=200}$	**19.18**	**19.28**	**19.32**	**19.20**	**19.29**	**19.22**	**19.17**	**19.12**
$RRS_{n=500}$	17.42	**18.06**	**17.89**	**17.75**	**17.73**	**17.62**	**17.56**	**17.43**
$RRS_{n=1000}$	16.53	**17.60**	**17.09**	**16.92**	**16.87**	**16.75**	**16.70**	**16.52**
$RRS_{n=10000}$	15.78	**17.35**	**16.53**	**16.30**	**16.19**	16.03	15.98	15.76
	Prec@10							
$RRS_{n=10}$	**24.43**	**24.95**	**24.01**	**24.20**	**24.26**	**24.33**	**24.55**	**24.48**
$RRS_{n=20}$	**24.31**	**23.86**	**23.60**	**23.94**	**24.06**	**24.06**	**24.16**	**24.16**
$RRS_{n=50}$	**25.04**	**24.91**	**25.31**	**25.23**	**25.20**	**25.13**	**25.32**	**25.04**
$RRS_{n=100}$	**28.63**	**26.11**	**28.26**	**28.04**	**28.37**	**28.01**	**28.42**	**28.71**
$RRS_{n=200}$	**30.07**	27.26	**29.27**	**29.26**	**29.59**	**29.59**	**30.00**	**30.14**
$RRS_{n=500}$	**30.29**	27.40	**28.84**	**29.55**	**30.02**	**29.59**	**30.29**	**30.29**
$RRS_{n=1000}$	**30.72**	27.98	**29.77**	**30.34**	**30.82**	**30.31**	**30.86**	**30.86**
$RRS_{n=10000}$	**31.44**	28.19	**29.92**	**30.77**	**31.25**	**30.46**	**31.44**	**31.44**
	rPrec							
$RRS_{n=10}$	**4.38**	**4.30**	**4.22**	**4.35**	**4.35**	**4.34**	**4.39**	**4.35**
$RRS_{n=20}$	**6.95**	6.45	**6.78**	**7.02**	**6.94**	**6.92**	**6.95**	**6.92**
$RRS_{n=50}$	**11.34**	9.96	10.80	**11.08**	**11.01**	**11.08**	**11.15**	**11.28**
$RRS_{n=100}$	**15.38**	13.13	14.26	14.61	**14.90**	**14.96**	**15.08**	**15.29**
$RRS_{n=200}$	**19.01**	15.26	17.39	18.13	18.35	**18.54**	**18.51**	**18.92**
$RRS_{n=500}$	**21.47**	15.98	19.33	19.73	**20.61**	**20.81**	**20.90**	**21.29**
$RRS_{n=1000}$	**21.70**	16.07	19.57	20.18	20.83	**21.00**	**20.92**	**21.48**
$RRS_{n=10000}$	**21.66**	16.04	19.80	20.36	20.79	**20.92**	**21.00**	**21.54**
	AvgPrec							
$RRS_{n=10}$	**2.59**	**2.55**	**2.56**	**2.60**	**2.55**	**2.56**	**2.60**	**2.59**
$RRS_{n=20}$	**3.73**	**3.48**	**3.74**	**3.80**	**3.67**	**3.67**	**3.69**	**3.71**
$RRS_{n=50}$	**6.12**	5.48	**5.89**	**5.93**	**5.94**	**5.94**	**5.98**	**6.09**
$RRS_{n=100}$	**8.56**	7.63	**8.24**	**8.28**	**8.29**	**8.34**	**8.40**	**8.52**
$RRS_{n=200}$	**11.36**	9.28	10.74	**10.86**	**11.01**	**11.07**	**11.11**	**11.28**
$RRS_{n=500}$	**14.58**	10.80	13.28	13.62	**14.03**	**14.21**	**14.29**	**14.52**
$RRS_{n=1000}$	**16.37**	11.52	14.69	15.12	15.62	15.87	**15.96**	**16.27**
$RRS_{n=10000}$	**17.58**	11.79	15.48	16.01	16.60	16.90	**17.06**	**17.44**

Table A.12: Impact of 2MA artist occurrence threshold on ANR (exalead, CAL500)

		unsupervised			*supervised*		*both*	
filter	none	ANR	2MA	A2	QB	QC	QB2	QC2
	Rec							
$RRS_{n=10}$	**2.18**	2.01	**2.07**	1.87	2.01	**2.89**	1.82	2.81
$RRS_{n=20}$	**3.74**	**3.95**	**3.93**	**3.74**	**3.89**	**4.50**	**3.83**	**4.21**
$RRS_{n=50}$	**7.17**	7.48	**7.87**	7.50	**8.15**	**8.81**	7.64	**8.53**
$RRS_{n=100}$	**12.72**	12.09	12.58	11.97	**12.80**	**14.42**	12.65	**12.65**
$RRS_{n=200}$	18.67	18.22	18.13	18.24	19.08	**21.80**	18.38	**19.85**
$RRS_{n=500}$	29.31	29.60	29.84	29.05	29.96	**32.34**	29.68	**31.79**
$RRS_{n=1000}$	40.38	40.31	40.11	39.78	40.43	**43.60**	39.83	41.40
$RRS_{n=10000}$	**80.50**	**79.56**	**76.80**	75.66	73.50	**76.02**	66.53	69.69
PseudoDoc	93.66	**93.31**	88.48	87.24	—	—	—	—
	Prec							
$RRS_{n=10}$	**30.15**	**31.89**	**35.72**	**31.63**	**34.32**	**34.97**	**34.56**	**32.70**
$RRS_{n=20}$	29.02	**31.30**	**32.86**	**31.18**	**33.91**	**33.78**	**34.10**	**32.78**
$RRS_{n=50}$	27.61	29.50	**31.13**	**30.01**	**32.50**	**33.32**	**32.32**	31.04
$RRS_{n=100}$	25.99	27.64	**29.05**	27.94	**31.75**	**30.65**	**31.57**	28.20
$RRS_{n=200}$	23.77	25.77	26.21	25.90	**28.60**	**28.39**	**28.59**	25.59
$RRS_{n=500}$	20.12	21.69	21.71	21.56	**24.76**	23.32	**24.67**	21.58
$RRS_{n=1000}$	16.88	17.86	18.19	17.90	**20.75**	19.14	**20.97**	18.05
$RRS_{n=10000}$	7.29	7.42	7.87	8.24	9.63	9.13	**12.52**	8.97
PseudoDoc	4.27	4.39	5.09	**5.54**	—	—	—	—
	Prec@10							
$RRS_{n=10}$	**31.19**	**34.94**	**37.76**	**33.23**	**36.45**	**35.72**	**35.34**	**34.16**
$RRS_{n=20}$	32.40	**34.96**	**37.05**	**33.19**	**36.75**	**37.38**	**36.36**	**33.84**
$RRS_{n=50}$	**38.45**	**36.20**	**41.70**	**38.00**	**40.25**	**40.05**	**40.15**	**38.05**
$RRS_{n=100}$	**44.10**	**39.05**	**46.65**	**40.30**	**43.40**	**45.00**	**43.10**	**41.60**
$RRS_{n=200}$	**47.75**	42.15	**48.90**	42.30	**46.95**	**49.20**	**47.25**	**45.25**
$RRS_{n=500}$	**50.30**	45.95	**51.20**	46.45	**49.75**	**51.05**	**50.35**	**50.15**
$RRS_{n=1000}$	**52.55**	48.75	**52.85**	**48.60**	**53.15**	**52.55**	**53.65**	**52.20**
$RRS_{n=10000}$	57.45	57.20	56.70	56.20	**62.35**	**61.25**	**62.40**	55.20
PseudoDoc	**39.25**	**41.05**	**40.50**	**41.65**	—	—	—	—
	rPrec							
$RRS_{n=10}$	**2.16**	1.99	**2.05**	1.85	2.01	**2.67**	1.82	2.12
$RRS_{n=20}$	**3.63**	3.68	**3.67**	**3.48**	**3.69**	**4.24**	**3.59**	**3.70**
$RRS_{n=50}$	**6.52**	6.85	**7.08**	6.75	7.30	**8.47**	7.16	**7.76**
$RRS_{n=100}$	**10.24**	10.41	**10.78**	10.49	**11.52**	**12.43**	10.97	**10.80**
$RRS_{n=200}$	14.22	14.54	14.68	14.75	16.03	**18.13**	15.66	15.74
$RRS_{n=500}$	19.84	21.01	20.35	21.02	22.54	**23.72**	22.37	**22.15**
$RRS_{n=1000}$	24.22	25.43	25.00	25.38	**27.48**	**28.55**	27.12	26.52
$RRS_{n=10000}$	35.20	**35.77**	35.03	34.97	**35.69**	**36.41**	34.53	32.56
PseudoDoc	30.78	**31.52**	31.03	**31.67**	—	—	—	—
	AvgPrec							
$RRS_{n=10}$	**1.19**	1.32	**1.38**	1.08	**1.39**	**1.78**	**1.15**	**1.44**
$RRS_{n=20}$	**1.84**	2.14	**2.24**	**1.93**	**2.26**	**2.70**	**2.15**	**2.23**
$RRS_{n=50}$	3.24	3.79	**4.06**	3.82	**4.49**	**5.14**	**4.22**	**4.42**
$RRS_{n=100}$	5.54	5.93	**6.29**	5.92	**7.00**	**7.71**	**6.75**	**6.29**
$RRS_{n=200}$	8.23	8.61	8.78	8.76	10.24	**11.64**	9.78	9.52
$RRS_{n=500}$	12.39	13.30	13.19	13.29	15.06	**15.90**	14.70	14.46
$RRS_{n=1000}$	16.10	17.37	17.01	17.27	19.41	**20.56**	19.23	18.27
$RRS_{n=10000}$	29.98	30.60	29.80	29.65	30.92	**31.64**	29.08	26.68
PseudoDoc	25.97	**26.64**	25.75	26.42	—	—	—	—

Table A.13: Comparison of filtering approaches (Google, c35k)

		unsupervised			*supervised*		*both*	
filter	none	ANR	2MA	A2	QB	QC	QB2	QC2
	Rec							
$RRS_{n=10}$	**1.73**	**1.60**	**1.90**	**1.69**	**1.61**	**1.49**	**1.67**	**2.02**
$RRS_{n=20}$	**2.80**	**2.82**	**3.05**	**2.97**	**2.88**	**3.03**	**2.96**	**2.68**
$RRS_{n=50}$	**5.46**	**5.88**	5.56	**6.30**	**5.97**	**6.22**	**6.27**	**5.83**
$RRS_{n=100}$	8.71	**9.57**	8.70	**9.60**	**9.91**	**10.30**	**10.00**	**10.44**
$RRS_{n=200}$	13.40	14.82	13.23	15.21	**15.81**	**17.56**	15.61	**15.66**
$RRS_{n=500}$	23.19	26.17	23.56	26.09	26.48	**28.55**	26.51	**26.84**
$RRS_{n=1000}$	34.01	36.30	33.92	35.63	36.07	**37.51**	34.65	**35.36**
$RRS_{n=10000}$	**72.48**	**72.30**	67.67	67.67	64.10	**67.27**	56.66	60.66
PseudoDoc	**87.20**	86.39	80.84	79.48	—	—	—	—
	Prec							
$RRS_{n=10}$	**28.42**	**28.45**	**28.00**	**28.50**	**29.86**	**29.77**	**29.79**	**27.95**
$RRS_{n=20}$	**28.05**	**25.94**	**27.48**	**25.92**	**27.25**	**28.46**	**26.69**	**28.49**
$RRS_{n=50}$	**26.43**	23.97	26.65	24.21	25.74	**30.09**	26.01	25.42
$RRS_{n=100}$	25.06	23.12	25.17	22.26	**25.36**	**29.14**	24.54	23.52
$RRS_{n=200}$	23.01	21.18	23.24	20.73	24.35	**27.75**	23.87	21.44
$RRS_{n=500}$	20.41	18.73	20.81	18.42	**21.61**	**22.67**	**21.55**	19.27
$RRS_{n=1000}$	17.97	16.65	18.39	16.26	**19.25**	**20.04**	**19.32**	16.45
$RRS_{n=10000}$	8.81	8.30	9.77	9.30	12.19	11.21	**14.99**	10.69
PseudoDoc	5.52	5.68	6.80	**7.11**	—	—	—	—
	Prec@10							
$RRS_{n=10}$	**27.40**	**29.53**	**26.66**	**29.18**	**30.88**	**32.22**	**31.01**	**31.85**
$RRS_{n=20}$	**30.30**	**29.58**	**29.19**	**29.16**	**31.11**	**33.48**	**31.09**	**32.05**
$RRS_{n=50}$	32.35	32.47	**33.27**	31.30	**35.77**	**39.45**	**33.95**	**33.75**
$RRS_{n=100}$	**35.90**	33.50	33.25	33.15	**36.05**	**41.65**	**35.90**	**37.10**
$RRS_{n=200}$	33.80	35.20	32.90	37.05	38.70	**44.50**	**39.60**	**39.75**
$RRS_{n=500}$	34.60	**39.55**	35.60	39.05	**43.15**	**45.75**	**43.80**	**40.50**
$RRS_{n=1000}$	36.80	41.95	36.85	40.10	**47.65**	**48.60**	**46.70**	**44.20**
$RRS_{n=10000}$	42.55	44.10	39.40	41.80	**50.45**	**55.60**	**49.65**	39.00
PseudoDoc	**34.95**	**35.75**	**32.70**	**33.70**	—	—	—	—
	rPrec							
$RRS_{n=10}$	**1.42**	**1.58**	**1.58**	**1.67**	**1.58**	**1.49**	**1.66**	**1.92**
$RRS_{n=20}$	**2.48**	**2.73**	**2.62**	**2.80**	**2.77**	**2.76**	**2.78**	**2.64**
$RRS_{n=50}$	**4.73**	**5.17**	4.88	**5.24**	**5.15**	**5.82**	**5.21**	**5.23**
$RRS_{n=100}$	7.41	7.96	7.73	7.88	8.00	**9.06**	**8.06**	**8.33**
$RRS_{n=200}$	10.76	11.50	10.49	11.43	**12.05**	**13.71**	11.87	11.62
$RRS_{n=500}$	15.70	16.76	16.38	16.71	18.41	**20.13**	18.17	17.79
$RRS_{n=1000}$	20.36	21.26	20.92	20.89	23.24	**25.01**	22.95	21.21
$RRS_{n=10000}$	**30.25**	**30.20**	28.69	28.24	**31.08**	**31.66**	29.17	26.09
PseudoDoc	26.72	**27.28**	25.60	26.42	—	—	—	—
	AvgPrec							
$RRS_{n=10}$	**0.93**	**0.82**	**0.96**	**0.88**	**0.85**	**0.78**	**0.90**	**1.12**
$RRS_{n=20}$	**1.38**	**1.29**	**1.44**	**1.39**	**1.41**	**1.59**	**1.46**	**1.41**
$RRS_{n=50}$	**2.42**	2.57	2.49	2.62	**2.86**	**3.18**	**2.80**	**2.59**
$RRS_{n=100}$	3.72	3.80	3.79	3.72	4.28	**4.98**	**4.29**	**4.12**
$RRS_{n=200}$	5.34	5.58	5.17	5.55	6.66	**7.88**	6.46	6.08
$RRS_{n=500}$	8.42	9.26	8.82	9.05	10.84	**12.21**	10.81	10.03
$RRS_{n=1000}$	11.96	12.97	12.37	12.61	15.04	**16.17**	14.58	13.05
$RRS_{n=10000}$	23.97	23.90	22.05	21.64	24.32	**25.55**	21.84	19.02
PseudoDoc	21.16	**21.76**	19.75	20.41	—	—	—	—

Table A.14: Comparison of filtering approaches (exalead, c35k)

		unsupervised			*supervised*		*both*	
filter	none	ANR	2MA	A2	QB	QC	QB2	QC2
	Rec							
$RRS_{n=10}$	**5.96**	**6.10**	**5.85**	**6.01**	**6.10**	**5.95**	**6.05**	**5.57**
$RRS_{n=20}$	**10.19**	**9.68**	**9.46**	**9.48**	**9.83**	**9.04**	**9.42**	**9.50**
$RRS_{n=50}$	**17.99**	**18.09**	**17.20**	**17.59**	**18.23**	**17.93**	**17.93**	15.92
$RRS_{n=100}$	**26.80**	**26.34**	**25.48**	**25.91**	**27.11**	22.65	**26.18**	20.51
$RRS_{n=200}$	**38.63**	**38.62**	**37.24**	**37.27**	**37.95**	28.61	**36.38**	22.23
$RRS_{n=500}$	**56.31**	**55.65**	**54.26**	**53.46**	51.21	45.49	46.89	42.93
$RRS_{n=1000}$	**66.91**	**66.48**	**63.78**	62.87	53.10	51.52	49.57	50.04
$RRS_{n=10000}$	**73.27**	**72.93**	69.06	68.38	37.98	45.21	36.56	45.61
PseudoDoc	**81.15**	**80.79**	77.95	77.24	—	—	—	—
	Prec							
$RRS_{n=10}$	**25.77**	**27.17**	**25.72**	**27.22**	**27.22**	**27.20**	**27.22**	**25.00**
$RRS_{n=20}$	**24.87**	**25.39**	**24.69**	**25.36**	**25.07**	**24.84**	**24.79**	**25.21**
$RRS_{n=50}$	**22.84**	**23.14**	**22.44**	**22.87**	**23.22**	**23.44**	**23.13**	**22.96**
$RRS_{n=100}$	**21.02**	**21.05**	**20.80**	**21.10**	**21.69**	**20.93**	**21.64**	**20.96**
$RRS_{n=200}$	**19.15**	**19.30**	**19.11**	**19.19**	**19.67**	**20.12**	**19.68**	**19.75**
$RRS_{n=500}$	16.86	16.95	16.92	17.00	**17.74**	**19.08**	**18.01**	**18.99**
$RRS_{n=1000}$	15.54	15.81	15.74	15.96	**17.36**	**17.92**	**18.65**	16.55
$RRS_{n=10000}$	14.56	14.84	14.82	15.13	**20.75**	**17.59**	**20.86**	16.75
PseudoDoc	14.50	14.55	14.82	**14.89**	—	—	—	—
	Prec@10							
$RRS_{n=10}$	**25.77**	**27.17**	**25.63**	**27.16**	**27.16**	**27.14**	**27.16**	**24.99**
$RRS_{n=20}$	**25.98**	**26.38**	**25.51**	**26.53**	**26.19**	**25.89**	**26.27**	**25.82**
$RRS_{n=50}$	**26.06**	**27.00**	**26.14**	**26.71**	**27.19**	**28.07**	**26.94**	**27.35**
$RRS_{n=100}$	**29.30**	**30.16**	**29.23**	**29.95**	**29.88**	**28.12**	**30.17**	**27.44**
$RRS_{n=200}$	**30.60**	**31.53**	**29.88**	**30.38**	**32.52**	**30.09**	**31.87**	27.71
$RRS_{n=500}$	**32.68**	**32.32**	**31.68**	**31.68**	**33.59**	**31.85**	**33.66**	**30.23**
$RRS_{n=1000}$	**33.47**	**32.75**	**32.61**	**31.97**	**33.56**	**34.02**	**33.69**	**31.52**
$RRS_{n=10000}$	**33.62**	**33.11**	**33.33**	**32.61**	**33.60**	**32.00**	**33.15**	**31.20**
PseudoDoc	**30.72**	**31.73**	**31.94**	**31.65**	—	—	—	—
	rPrec							
$RRS_{n=10}$	**5.61**	**6.00**	**5.50**	**5.90**	**6.10**	**5.81**	**6.05**	**5.36**
$RRS_{n=20}$	**8.84**	**8.68**	**8.54**	**8.55**	**8.90**	**8.48**	**8.84**	**8.24**
$RRS_{n=50}$	**13.49**	**13.77**	**13.08**	**13.35**	**13.58**	**13.06**	**13.34**	**12.52**
$RRS_{n=100}$	**18.05**	**17.83**	**17.81**	**17.43**	**18.21**	14.84	**17.85**	13.21
$RRS_{n=200}$	**21.58**	**22.05**	**21.14**	**21.78**	**21.92**	17.57	**21.19**	14.50
$RRS_{n=500}$	**24.06**	**24.69**	**23.59**	**24.01**	**23.69**	21.28	**22.53**	20.14
$RRS_{n=1000}$	**24.86**	**25.43**	**24.54**	**24.69**	**23.36**	**22.96**	**22.63**	21.89
$RRS_{n=10000}$	**25.06**	**25.60**	**24.96**	**24.85**	18.91	19.85	18.00	19.71
PseudoDoc	**25.77**	**25.76**	**25.49**	**25.68**	—	—	—	—
	AvgPrec							
$RRS_{n=10}$	**3.58**	**3.46**	**3.41**	**3.44**	**3.52**	**3.68**	**3.47**	**3.60**
$RRS_{n=20}$	**5.30**	**4.88**	**5.11**	**4.82**	**5.00**	**5.00**	**4.88**	**5.04**
$RRS_{n=50}$	**7.57**	**8.03**	**7.34**	**7.82**	**7.99**	**8.35**	**7.87**	**7.36**
$RRS_{n=100}$	**10.59**	**10.96**	**10.18**	**10.65**	**10.84**	9.12	**10.62**	8.34
$RRS_{n=200}$	**13.84**	**14.02**	**13.24**	**13.59**	**14.12**	10.86	**13.71**	9.22
$RRS_{n=500}$	**18.02**	**18.20**	**17.32**	**17.36**	**17.24**	15.55	16.41	14.71
$RRS_{n=1000}$	**20.37**	**20.42**	**19.48**	**19.35**	18.22	17.81	17.16	16.64
$RRS_{n=10000}$	**21.77**	**22.04**	**20.75**	**20.74**	13.96	15.22	13.55	14.93
PseudoDoc	**22.66**	**22.84**	21.81	21.93	—	—	—	—

Table A.15: Comparison of filtering approaches (Google, CAL500)

		unsupervised			*supervised*		*both*	
filter	none	ANR	2MA	A2	QB	QC	QB2	QC2
	Rec							
$RRS_{n=10}$	**5.58**	**4.65**	**5.42**	**4.62**	**4.93**	3.94	**4.87**	4.27
$RRS_{n=20}$	**9.13**	**8.46**	**8.39**	**8.24**	**8.83**	6.56	**8.58**	6.83
$RRS_{n=50}$	**15.68**	**15.63**	**15.21**	**15.10**	**16.02**	13.23	**16.23**	13.01
$RRS_{n=100}$	**23.66**	**23.92**	**22.86**	**23.25**	**24.08**	17.51	**23.70**	14.86
$RRS_{n=200}$	**33.42**	**34.26**	**32.34**	**32.88**	**31.75**	22.64	30.67	18.18
$RRS_{n=500}$	**47.36**	**47.22**	**45.40**	45.39	39.67	34.22	37.34	29.94
$RRS_{n=1000}$	**56.09**	**55.68**	**53.39**	52.67	38.96	36.95	36.63	35.01
$RRS_{n=10000}$	**61.43**	**60.28**	58.17	56.71	27.05	30.06	26.00	27.31
$PseudoDoc$	**70.52**	**69.42**	67.09	66.02	—	—	—	—
	Prec							
$RRS_{n=10}$	**26.74**	**24.40**	**26.73**	**24.18**	**24.22**	19.95	**23.95**	**21.25**
$RRS_{n=20}$	**24.11**	**23.97**	**24.30**	**23.74**	**23.39**	**21.67**	**23.12**	**21.78**
$RRS_{n=50}$	**22.11**	**21.87**	**22.23**	**21.69**	**21.65**	**20.83**	**22.61**	**20.46**
$RRS_{n=100}$	**20.63**	**20.77**	**20.73**	**20.89**	**20.73**	**20.67**	**21.19**	**20.41**
$RRS_{n=200}$	**18.72**	**19.18**	**19.16**	**19.29**	**19.19**	**20.71**	**19.17**	**19.53**
$RRS_{n=500}$	17.20	17.42	17.62	17.73	**18.01**	**19.36**	**18.22**	**19.21**
$RRS_{n=1000}$	16.25	16.53	16.53	16.87	**18.64**	**18.61**	**19.52**	18.03
$RRS_{n=10000}$	15.45	15.78	15.82	16.19	**21.08**	**21.44**	**20.88**	17.16
$PseudoDoc$	15.08	15.13	15.31	**15.37**	—	—	—	—
	Prec@10							
$RRS_{n=10}$	**26.80**	**24.43**	**26.62**	**24.26**	**24.34**	19.95	**23.97**	**21.26**
$RRS_{n=20}$	**25.42**	**24.31**	**25.15**	**24.06**	**23.84**	**21.32**	**23.37**	**22.52**
$RRS_{n=50}$	**26.55**	**25.04**	**26.21**	**25.20**	**25.11**	**25.06**	**25.51**	**22.33**
$RRS_{n=100}$	**27.99**	**28.63**	**26.57**	**28.37**	**27.29**	**26.23**	**27.70**	**25.12**
$RRS_{n=200}$	**28.63**	**30.07**	**28.23**	**29.59**	**29.10**	**28.30**	**27.38**	**25.36**
$RRS_{n=500}$	**29.28**	**30.29**	**28.80**	**30.02**	**30.20**	**29.26**	**29.63**	**31.07**
$RRS_{n=1000}$	**29.50**	**30.72**	**29.88**	**30.82**	**31.37**	**29.71**	**31.98**	**29.76**
$RRS_{n=10000}$	**30.43**	**31.44**	**30.17**	**31.25**	**30.38**	**29.98**	**29.09**	26.03
$PseudoDoc$	**26.47**	**27.84**	**25.61**	**27.55**	—	—	—	—
	rPrec							
$RRS_{n=10}$	**5.33**	**4.38**	**5.26**	**4.35**	**4.58**	3.73	**4.61**	4.27
$RRS_{n=20}$	**7.65**	**6.95**	**7.57**	**6.94**	**7.25**	6.15	**7.15**	6.46
$RRS_{n=50}$	**11.88**	**11.34**	**11.63**	**11.01**	**11.81**	10.10	**11.59**	9.81
$RRS_{n=100}$	**15.81**	**15.38**	**15.46**	**14.90**	**15.53**	12.32	**14.92**	10.96
$RRS_{n=200}$	**18.77**	**19.01**	**18.31**	**18.35**	**18.60**	14.35	**18.18**	12.47
$RRS_{n=500}$	**20.77**	**21.47**	**20.09**	**20.61**	**19.33**	17.96	18.36	16.32
$RRS_{n=1000}$	**21.02**	**21.70**	**20.79**	**20.83**	18.62	17.93	17.96	17.46
$RRS_{n=10000}$	**21.12**	**21.66**	**20.53**	**20.79**	15.28	15.51	14.76	14.14
$PseudoDoc$	**21.31**	**22.39**	**20.67**	**22.03**	—	—	—	—
	AvgPrec							
$RRS_{n=10}$	**2.78**	**2.59**	**2.92**	**2.55**	**2.73**	2.26	**2.67**	**2.54**
$RRS_{n=20}$	**4.16**	**3.73**	**4.16**	**3.67**	**3.95**	3.53	**3.85**	**3.56**
$RRS_{n=50}$	**6.34**	**6.12**	**6.12**	**5.94**	**6.37**	5.73	**6.29**	5.55
$RRS_{n=100}$	**8.52**	**8.56**	**8.24**	**8.29**	**8.74**	7.22	**8.71**	6.49
$RRS_{n=200}$	**10.85**	**11.36**	**10.55**	**11.01**	**11.12**	8.93	**10.72**	7.53
$RRS_{n=500}$	**14.15**	**14.58**	**13.62**	**14.03**	13.25	11.94	12.69	11.07
$RRS_{n=1000}$	**15.93**	**16.37**	**15.12**	**15.62**	13.44	12.74	13.12	12.24
$RRS_{n=10000}$	**16.96**	**17.58**	15.94	16.60	10.44	10.80	10.06	9.83
$PseudoDoc$	**17.57**	**18.57**	16.53	17.60	—	—	—	—

Table A.16: Comparison of filtering approaches (exalead, CAL500)

A.2.3 Audio-Based Re-Ranking Impact

PAR, α=10	$k=0$	$k=10$	$k=25$	$k=50$	$k=0$	$k=10$	$k=25$	$k=50$
	Rec				**Prec**			
$RRS_{n=10}$	2.18	4.54	7.28	**10.71**	**30.15**	9.30	7.64	6.81
$RRS_{n=20}$	3.74	7.65	11.85	**16.89**	**29.02**	9.12	7.43	6.57
$RRS_{n=50}$	7.17	14.04	20.56	**27.76**	**27.61**	8.73	7.06	6.17
$RRS_{n=100}$	12.72	22.73	30.85	**39.41**	**25.99**	8.18	6.51	5.66
$RRS_{n=200}$	18.67	31.86	41.69	**50.98**	**23.77**	7.56	6.02	5.25
$RRS_{n=500}$	29.31	46.90	57.56	**66.60**	**20.12**	6.81	5.47	4.81
$RRS_{n=1000}$	40.38	60.12	70.06	**77.63**	**16.88**	6.15	5.03	4.50
$RRS_{n=10000}$	80.50	92.26	95.17	**96.68**	**7.29**	4.32	3.99	3.85
PseudoDoc	93.66	97.06	98.04	**98.50**	**4.27**	3.75	3.70	3.68
	Prec@10				**rPrec**			
$RRS_{n=10}$	**31.19**	30.85	30.85	30.85	2.16	3.91	5.15	**6.22**
$RRS_{n=20}$	**32.40**	**32.15**	**32.15**	**32.15**	3.63	6.06	**7.48**	**8.46**
$RRS_{n=50}$	**38.45**	**38.20**	**38.20**	**38.20**	6.52	9.94	**11.15**	**11.67**
$RRS_{n=100}$	**44.10**	**44.15**	**44.15**	**44.15**	10.24	**13.94**	**14.64**	**14.75**
$RRS_{n=200}$	**47.75**	**46.35**	**46.35**	**46.35**	14.22	**17.69**	**17.85**	**17.86**
$RRS_{n=500}$	**50.30**	**47.45**	**47.45**	**47.45**	19.84	**22.04**	**22.04**	**22.04**
$RRS_{n=1000}$	**52.55**	43.00	43.00	43.00	24.22	**24.87**	**24.87**	**24.87**
$RRS_{n=10000}$	**57.45**	17.50	17.50	17.50	**35.20**	22.41	22.41	22.41
PseudoDoc	**39.25**	8.55	8.55	8.55	**30.78**	12.57	12.57	12.57
	AvgPrec				**AUC**			
$RRS_{n=10}$	1.19	1.61	1.90	**2.21**	3.05	3.28	3.48	**3.77**
$RRS_{n=20}$	1.84	2.53	2.95	**3.38**	3.64	4.04	4.48	**4.90**
$RRS_{n=50}$	3.24	4.37	4.99	**5.54**	4.99	5.92	6.55	**7.13**
$RRS_{n=100}$	5.54	7.12	7.83	**8.42**	7.11	8.61	9.37	**9.93**
$RRS_{n=200}$	8.23	10.11	10.88	**11.47**	9.62	11.48	12.25	**12.83**
$RRS_{n=500}$	12.39	14.25	15.00	**15.51**	13.76	15.68	16.38	**16.92**
$RRS_{n=1000}$	16.10	17.06	17.70	**18.10**	17.22	18.30	18.94	**19.36**
$RRS_{n=10000}$	**29.98**	17.82	17.95	**18.02**	**31.25**	19.42	19.53	19.55
PseudoDoc	**25.97**	11.70	11.71	11.72	**27.09**	12.93	12.96	12.98

Table A.17: Impact of k on PAR with α=10 (Google, c35k)

PAR, α=100	$k=0$	$k=10$	$k=25$	$k=50$	$k=0$	$k=10$	$k=25$	$k=50$
	Rec				**Prec**			
$RRS_{n=10}$	2.18	4.54	7.28	**10.71**	**30.15**	9.30	7.64	6.81
$RRS_{n=20}$	3.74	7.65	11.85	**16.89**	**29.02**	9.12	7.43	6.57
$RRS_{n=50}$	7.17	14.04	20.56	**27.76**	**27.61**	8.73	7.06	6.17
$RRS_{n=100}$	12.72	22.73	30.85	**39.41**	**25.99**	8.18	6.51	5.66
$RRS_{n=200}$	18.67	31.86	41.69	**50.98**	**23.77**	7.56	6.02	5.25
$RRS_{n=500}$	29.31	46.90	57.56	**66.60**	**20.12**	6.81	5.47	4.81
$RRS_{n=1000}$	40.38	60.12	70.06	**77.63**	**16.88**	6.15	5.03	4.50
$RRS_{n=10000}$	80.50	92.26	95.17	**96.68**	**7.29**	4.32	3.99	3.85
PseudoDoc	93.66	97.06	98.04	**98.50**	**4.27**	3.75	3.70	3.68
	Prec@10				**rPrec**			
$RRS_{n=10}$	**31.19**	30.85	30.85	30.85	2.16	3.91	5.15	**6.22**
$RRS_{n=20}$	**32.40**	32.40	32.40	32.40	3.63	6.06	**7.48**	**8.46**
$RRS_{n=50}$	**38.45**	38.45	38.45	38.45	6.52	9.94	**11.15**	**11.67**
$RRS_{n=100}$	**44.10**	**44.05**	**44.05**	**44.05**	10.24	**13.93**	**14.63**	**14.74**
$RRS_{n=200}$	**47.75**	**47.75**	**47.75**	**47.75**	14.22	**17.67**	**17.83**	**17.84**
$RRS_{n=500}$	**50.30**	**50.35**	**50.35**	**50.35**	19.84	**22.04**	**22.04**	**22.04**
$RRS_{n=1000}$	**52.55**	**52.30**	**52.30**	**52.30**	24.22	**25.24**	**25.24**	**25.24**
$RRS_{n=10000}$	**57.45**	41.25	41.25	41.25	**35.20**	33.12	33.12	33.12
PseudoDoc	**39.25**	23.40	23.40	23.40	**30.78**	26.22	26.22	26.22
	AvgPrec				**AUC**			
$RRS_{n=10}$	1.19	1.61	1.90	**2.21**	3.05	3.28	3.48	**3.77**
$RRS_{n=20}$	1.84	2.53	2.95	**3.38**	3.64	4.05	4.50	**4.91**
$RRS_{n=50}$	3.24	4.38	4.99	**5.54**	4.99	5.94	6.57	**7.15**
$RRS_{n=100}$	5.54	7.14	7.85	**8.45**	7.11	8.67	9.43	**10.00**
$RRS_{n=200}$	8.23	10.17	10.94	**11.53**	9.62	11.57	12.34	**12.93**
$RRS_{n=500}$	12.39	14.69	15.44	**15.95**	13.76	16.19	16.89	**17.42**
$RRS_{n=1000}$	16.10	18.24	18.88	**19.28**	17.22	19.47	20.11	**20.52**
$RRS_{n=10000}$	29.98	27.14	27.27	**27.34**	**31.25**	28.52	28.63	**28.66**
PseudoDoc	**25.97**	21.38	21.40	21.41	**27.09**	22.63	22.65	22.67

Table A.18: Impact of k on PAR with α=100 (Google, c35k)

PAR, α=10	$k=0$	$k=10$	$k=25$	$k=50$	$k=0$	$k=10$	$k=25$	$k=50$
	Rec				**Prec**			
$RRS_{n=10}$	1.73	3.34	5.23	**7.76**	**28.42**	9.03	7.38	6.64
$RRS_{n=20}$	2.80	5.48	8.51	**12.31**	**28.05**	8.94	7.50	6.81
$RRS_{n=50}$	5.46	10.94	16.39	**22.59**	**26.43**	8.62	7.02	6.20
$RRS_{n=100}$	8.71	17.22	24.74	**32.73**	**25.06**	8.14	6.58	5.79
$RRS_{n=200}$	13.40	25.52	34.86	**44.03**	**23.01**	7.65	6.18	5.42
$RRS_{n=500}$	23.19	40.01	50.75	**60.14**	**20.41**	7.03	5.68	4.99
$RRS_{n=1000}$	34.01	53.43	63.92	**72.12**	**17.97**	6.44	5.24	4.66
$RRS_{n=10000}$	72.48	86.97	91.12	**93.66**	**8.81**	4.54	4.10	3.91
PseudoDoc	87.20	93.09	94.92	**96.09**	**5.52**	3.94	3.79	3.73
	Prec@10				**rPrec**			
$RRS_{n=10}$	**27.40**	26.95	26.95	26.95	1.42	2.83	3.94	**4.94**
$RRS_{n=20}$	**30.30**	**30.35**	**30.35**	**30.35**	2.48	4.55	5.89	**6.92**
$RRS_{n=50}$	**32.35**	**32.25**	**32.25**	**32.25**	4.73	7.96	**9.25**	**9.89**
$RRS_{n=100}$	**35.90**	**35.55**	**35.55**	**35.55**	7.41	11.13	**11.97**	**12.21**
$RRS_{n=200}$	**33.80**	**34.65**	**34.65**	**34.65**	10.76	**14.53**	**14.83**	**14.90**
$RRS_{n=500}$	**34.60**	**36.00**	**36.00**	**36.00**	15.70	**18.41**	**18.50**	**18.52**
$RRS_{n=1000}$	**36.80**	**36.25**	**36.25**	**36.25**	20.36	**21.81**	**21.85**	**21.85**
$RRS_{n=10000}$	**42.55**	23.00	23.00	23.00	**30.25**	24.40	24.40	24.40
PseudoDoc	**34.95**	12.00	12.00	12.00	**26.72**	15.26	15.26	15.26
	AvgPrec				**AUC**			
$RRS_{n=10}$	0.93	1.29	1.53	**1.79**	2.79	2.95	3.12	**3.36**
$RRS_{n=20}$	1.38	1.91	2.23	**2.57**	3.13	3.43	3.74	**4.06**
$RRS_{n=50}$	2.42	3.34	3.85	**4.34**	4.13	4.77	5.23	**5.74**
$RRS_{n=100}$	3.72	5.02	5.67	**6.25**	5.17	6.36	7.03	**7.65**
$RRS_{n=200}$	5.34	7.06	7.83	**8.44**	6.62	8.29	9.09	**9.71**
$RRS_{n=500}$	8.42	10.48	11.27	**11.83**	9.79	11.75	12.54	**13.09**
$RRS_{n=1000}$	11.96	13.94	14.65	**15.12**	13.36	15.34	16.04	**16.50**
$RRS_{n=10000}$	23.97	18.77	18.96	**19.06**	**25.32**	20.33	20.50	20.58
PseudoDoc	**21.16**	12.69	12.74	12.77	**22.29**	13.88	13.93	13.95

Table A.19: Impact of k on PAR with α=10 (exalead, c35k)

PAR, α=100	$k=0$	$k=10$	$k=25$	$k=50$	$k=0$	$k=10$	$k=25$	$k=50$
	Rec				**Prec**			
$RRS_{n=10}$	1.73	3.34	5.23	**7.76**	**28.42**	9.03	7.38	6.64
$RRS_{n=20}$	2.80	5.48	8.51	**12.31**	**28.05**	8.94	7.50	6.81
$RRS_{n=50}$	5.46	10.94	16.39	**22.59**	**26.43**	8.62	7.02	6.20
$RRS_{n=100}$	8.71	17.22	24.74	**32.73**	**25.06**	8.14	6.58	5.79
$RRS_{n=200}$	13.40	25.52	34.86	**44.03**	**23.01**	7.65	6.18	5.42
$RRS_{n=500}$	23.19	40.01	50.75	**60.14**	**20.41**	7.03	5.68	4.99
$RRS_{n=1000}$	34.01	53.43	63.92	**72.12**	**17.97**	6.44	5.24	4.66
$RRS_{n=10000}$	72.48	86.97	91.12	**93.66**	**8.81**	4.54	4.10	3.91
$PseudoDoc$	87.20	93.09	94.92	**96.09**	**5.52**	3.94	3.79	3.73
	Prec@10				**rPrec**			
$RRS_{n=10}$	**27.40**	26.95	26.95	26.95	1.42	2.83	3.94	**4.94**
$RRS_{n=20}$	**30.30**	**30.35**	**30.35**	**30.35**	2.48	4.55	5.89	**6.92**
$RRS_{n=50}$	**32.35**	32.35	32.35	32.35	4.73	7.96	**9.25**	**9.89**
$RRS_{n=100}$	**35.90**	**35.85**	**35.85**	**35.85**	7.41	11.13	**11.97**	**12.21**
$RRS_{n=200}$	**33.80**	**33.95**	**33.95**	**33.95**	10.76	**14.54**	**14.84**	**14.92**
$RRS_{n=500}$	**34.60**	**34.95**	**34.95**	**34.95**	15.70	**18.42**	**18.50**	**18.52**
$RRS_{n=1000}$	**36.80**	**37.25**	**37.25**	**37.25**	20.36	**21.94**	**21.99**	**21.99**
$RRS_{n=10000}$	**42.55**	**38.45**	**38.45**	**38.45**	**30.25**	**29.55**	**29.55**	**29.55**
$PseudoDoc$	**34.95**	26.65	26.65	26.65	**26.72**	24.86	24.86	24.86
	AvgPrec				**AUC**			
$RRS_{n=10}$	0.93	1.29	1.53	**1.79**	2.79	2.95	3.12	**3.36**
$RRS_{n=20}$	1.38	1.91	2.23	**2.57**	3.13	3.43	3.75	**4.06**
$RRS_{n=50}$	2.42	3.34	3.84	**4.34**	4.13	4.78	5.24	**5.75**
$RRS_{n=100}$	3.72	5.04	5.69	**6.27**	5.17	6.41	7.08	**7.70**
$RRS_{n=200}$	5.34	7.10	7.86	**8.47**	6.62	8.34	9.14	**9.76**
$RRS_{n=500}$	8.42	10.61	11.39	**11.95**	9.79	11.92	12.70	**13.25**
$RRS_{n=1000}$	11.96	14.28	14.99	**15.45**	13.36	15.73	16.44	**16.89**
$RRS_{n=10000}$	23.97	23.84	24.04	**24.14**	25.32	25.22	**25.39**	**25.46**
$PseudoDoc$	**21.16**	19.41	19.46	**19.49**	**22.29**	20.55	20.60	20.62

Table A.20: Impact of k on PAR with α=100 (exalead, c35k)

PAR, α=10	$k=0$	$k=10$	$k=25$	$k=50$	$k=0$	$k=10$	$k=25$	$k=50$
	Rec				**Prec**			
$RRS_{n=10}$	5.96	23.14	40.17	**59.08**	**25.77**	**16.53**	15.56	14.84
$RRS_{n=20}$	10.19	36.98	58.12	**75.90**	**24.87**	**15.96**	14.98	14.34
$RRS_{n=50}$	17.99	58.37	79.89	**89.52**	**22.84**	15.09	14.16	13.63
$RRS_{n=100}$	26.80	74.95	89.18	**94.78**	**21.02**	14.48	13.66	13.39
$RRS_{n=200}$	38.63	84.55	93.79	**96.38**	**19.15**	13.83	13.39	13.22
$RRS_{n=500}$	56.31	91.02	**95.63**	**97.05**	**16.86**	13.42	13.24	13.15
$RRS_{n=1000}$	66.91	92.14	**95.85**	**97.18**	**15.54**	13.28	13.19	13.13
$RRS_{n=10000}$	73.27	92.40	**96.01**	**97.31**	**14.56**	13.23	13.18	13.13
$PseudoDoc$	81.15	95.93	**98.21**	**98.83**	**14.50**	13.33	13.36	13.34
	Prec@10				**rPrec**			
$RRS_{n=10}$	**25.77**	23.74	23.74	23.74	5.61	13.95	**17.39**	**18.44**
$RRS_{n=20}$	**25.98**	**25.61**	**25.61**	**25.61**	8.84	**18.80**	**20.35**	**20.70**
$RRS_{n=50}$	**26.06**	**26.04**	**26.04**	**26.04**	13.49	**21.74**	**22.20**	**22.20**
$RRS_{n=100}$	**29.30**	**29.50**	**29.50**	**29.50**	18.05	**23.54**	**23.67**	**23.67**
$RRS_{n=200}$	**30.60**	**30.94**	**30.94**	**30.94**	21.58	**24.25**	**24.38**	**24.39**
$RRS_{n=500}$	**32.68**	**33.02**	**33.02**	**33.02**	24.06	**25.69**	**25.82**	**25.83**
$RRS_{n=1000}$	**33.47**	**31.44**	**31.44**	**31.44**	**24.86**	**24.76**	**24.89**	**24.89**
$RRS_{n=10000}$	**33.62**	30.29	30.29	30.29	**25.06**	**24.30**	**24.43**	**24.44**
$PseudoDoc$	**30.72**	27.63	27.63	27.63	**25.77**	**24.15**	**24.15**	**24.15**
	AvgPrec				**AUC**			
$RRS_{n=10}$	3.58	7.38	10.46	**13.51**	4.50	8.33	11.56	**14.77**
$RRS_{n=20}$	5.30	10.65	14.35	**17.20**	6.13	11.72	15.72	**18.64**
$RRS_{n=50}$	7.57	15.21	18.58	**20.16**	8.63	16.61	20.12	**21.65**
$RRS_{n=100}$	10.59	19.42	21.64	**22.44**	11.69	20.86	23.26	**23.73**
$RRS_{n=200}$	13.84	22.36	23.68	**24.09**	15.16	23.97	**25.12**	**25.44**
$RRS_{n=500}$	18.02	23.80	**24.58**	**24.84**	19.33	25.18	**25.91**	**26.18**
$RRS_{n=1000}$	20.37	23.59	**24.24**	**24.49**	21.77	24.89	**25.57**	**25.84**
$RRS_{n=10000}$	21.77	23.03	**23.66**	**23.91**	23.16	24.47	**25.14**	**25.39**
$PseudoDoc$	22.66	23.44	**23.77**	**23.89**	24.12	**24.86**	**25.20**	**25.36**

Table A.21: Impact of k on PAR with α=10 (Google, CAL500)

PAR, α=100	$k=0$	$k=10$	$k=25$	$k=50$	$k=0$	$k=10$	$k=25$	$k=50$
	Rec				**Prec**			
$RRS_{n=10}$	5.96	23.14	40.17	**59.08**	**25.77**	**16.53**	15.56	14.84
$RRS_{n=20}$	10.19	36.98	58.12	**75.90**	**24.87**	**15.96**	14.98	14.34
$RRS_{n=50}$	17.99	58.37	79.89	**89.52**	**22.84**	15.09	14.16	13.63
$RRS_{n=100}$	26.80	74.95	89.18	**94.78**	**21.02**	14.48	13.66	13.39
$RRS_{n=200}$	38.63	84.55	93.79	**96.38**	**19.15**	13.83	13.39	13.22
$RRS_{n=500}$	56.31	91.02	**95.63**	**97.05**	**16.86**	13.42	13.24	13.15
$RRS_{n=1000}$	66.91	92.14	**95.85**	**97.18**	**15.54**	13.28	13.19	13.13
$RRS_{n=10000}$	73.27	92.40	**96.01**	**97.31**	**14.56**	13.23	13.18	13.13
$PseudoDoc$	81.15	95.93	**98.21**	**98.83**	**14.50**	13.33	13.36	13.34
	Prec@10				**rPrec**			
$RRS_{n=10}$	**25.77**	23.74	23.74	23.74	5.61	13.95	**17.39**	**18.44**
$RRS_{n=20}$	**25.98**	**25.61**	**25.61**	**25.61**	8.84	**18.80**	**20.35**	**20.70**
$RRS_{n=50}$	**26.06**	**26.04**	**26.04**	**26.04**	13.49	**21.76**	**22.22**	**22.23**
$RRS_{n=100}$	**29.30**	**29.28**	**29.28**	**29.28**	18.05	**23.65**	**23.78**	**23.79**
$RRS_{n=200}$	**30.60**	**30.58**	**30.58**	**30.58**	21.58	**24.25**	**24.38**	**24.38**
$RRS_{n=500}$	**32.68**	**32.88**	**32.88**	**32.88**	24.06	**25.75**	**25.88**	**25.89**
$RRS_{n=1000}$	**33.47**	**33.74**	**33.74**	**33.74**	24.86	**26.33**	**26.46**	**26.46**
$RRS_{n=10000}$	**33.62**	**33.38**	**33.38**	**33.38**	25.06	**26.68**	**26.81**	**26.81**
$PseudoDoc$	**30.72**	**31.01**	**31.01**	**31.01**	**25.77**	**26.64**	**26.64**	**26.64**
	AvgPrec				**AUC**			
$RRS_{n=10}$	3.58	7.38	10.46	**13.51**	4.50	8.33	11.56	**14.77**
$RRS_{n=20}$	5.30	10.65	14.35	**17.20**	6.13	11.72	15.72	**18.64**
$RRS_{n=50}$	7.57	15.23	18.61	**20.19**	8.63	16.65	20.16	**21.69**
$RRS_{n=100}$	10.59	19.39	21.61	**22.41**	11.69	20.82	23.23	**23.70**
$RRS_{n=200}$	13.84	22.16	23.48	**23.90**	15.16	23.81	**24.95**	**25.28**
$RRS_{n=500}$	18.02	24.20	**24.99**	**25.25**	19.33	25.68	**26.41**	**26.68**
$RRS_{n=1000}$	20.37	25.02	**25.67**	**25.92**	21.77	26.39	**27.07**	**27.34**
$RRS_{n=10000}$	21.77	25.31	**25.93**	**26.18**	23.16	26.68	**27.35**	**27.59**
$PseudoDoc$	22.66	25.44	**25.78**	**25.89**	24.12	**26.91**	**27.25**	**27.41**

Table A.22: Impact of k on PAR with α=100 (Google, CAL500)

PAR, α=10	$k=0$	$k=10$	$k=25$	$k=50$	$k=0$	$k=10$	$k=25$	$k=50$
	Rec				**Prec**			
$RRS_{n=10}$	5.58	23.53	42.03	**61.04**	**26.74**	**16.94**	15.77	14.87
$RRS_{n=20}$	9.13	35.39	58.57	**75.70**	**24.11**	15.80	14.88	14.25
$RRS_{n=50}$	15.68	56.68	78.49	**90.02**	**22.11**	**15.37**	14.27	13.76
$RRS_{n=100}$	23.66	71.86	87.31	**94.04**	**20.63**	14.67	13.85	13.47
$RRS_{n=200}$	33.42	81.96	91.28	**95.13**	**18.72**	14.08	13.59	13.33
$RRS_{n=500}$	47.36	86.55	**92.94**	**95.59**	**17.20**	13.75	13.46	13.27
$RRS_{n=1000}$	56.09	87.46	**93.24**	**95.85**	**16.25**	13.64	13.43	13.26
$RRS_{n=10000}$	61.43	87.68	**93.29**	**95.87**	**15.45**	13.59	13.40	13.25
PseudoDoc	70.52	92.90	**96.30**	**97.97**	**15.08**	13.53	13.43	13.38
	Prec@10				**rPrec**			
$RRS_{n=10}$	**26.80**	24.24	24.24	24.24	5.33	14.86	**17.78**	**18.74**
$RRS_{n=20}$	**25.42**	**24.24**	**24.24**	**24.24**	7.65	**18.04**	**19.32**	**19.74**
$RRS_{n=50}$	**26.55**	**26.04**	**26.04**	**26.04**	11.88	**20.38**	**20.88**	**21.01**
$RRS_{n=100}$	**27.99**	**27.27**	**27.27**	**27.27**	15.81	**21.87**	**22.18**	**22.30**
$RRS_{n=200}$	**28.63**	**27.84**	**27.84**	**27.84**	18.77	**22.37**	**22.67**	**22.80**
$RRS_{n=500}$	**29.28**	**29.21**	**29.21**	**29.21**	**20.77**	**23.09**	**23.39**	**23.51**
$RRS_{n=1000}$	**29.50**	**28.99**	**28.99**	**28.99**	21.02	**22.90**	**23.20**	**23.33**
$RRS_{n=10000}$	**30.43**	27.77	27.77	27.77	**21.12**	**21.89**	**22.19**	**22.32**
PseudoDoc	**26.47**	**26.33**	**26.33**	**26.33**	**21.31**	**22.30**	**22.41**	**22.47**
	AvgPrec				**AUC**			
$RRS_{n=10}$	2.78	7.00	10.24	**13.33**	3.77	8.03	11.43	**14.61**
$RRS_{n=20}$	4.16	9.62	13.32	**16.06**	5.00	10.77	14.63	**17.55**
$RRS_{n=50}$	6.34	14.17	17.49	**19.12**	7.22	15.58	19.05	**20.66**
$RRS_{n=100}$	8.52	17.25	19.70	**20.63**	9.61	18.71	21.33	**22.04**
$RRS_{n=200}$	10.85	19.29	20.93	**21.54**	12.13	20.73	22.30	**22.80**
$RRS_{n=500}$	14.15	20.75	**21.90**	**22.36**	15.51	22.06	**23.23**	**23.70**
$RRS_{n=1000}$	15.93	20.94	**22.01**	**22.46**	17.35	22.18	**23.36**	**23.82**
$RRS_{n=10000}$	16.96	20.00	**21.05**	**21.50**	18.27	21.37	**22.51**	**22.96**
PseudoDoc	17.57	21.24	**21.84**	**22.07**	19.21	22.66	**23.28**	**23.54**

Table A.23: Impact of k on PAR with α=10 (exalead, CAL500)

PAR, α=100	$k=0$	$k=10$	$k=25$	$k=50$	$k=0$	$k=10$	$k=25$	$k=50$
	Rec				**Prec**			
$RRS_{n=10}$	5.58	23.53	42.03	**61.04**	**26.74**	**16.94**	15.77	14.87
$RRS_{n=20}$	9.13	35.39	58.57	**75.70**	**24.11**	15.80	14.88	14.25
$RRS_{n=50}$	15.68	56.68	78.49	**90.02**	**22.11**	**15.37**	14.27	13.76
$RRS_{n=100}$	23.66	71.86	87.31	**94.04**	**20.63**	14.67	13.85	13.47
$RRS_{n=200}$	33.42	81.96	91.28	**95.13**	**18.72**	14.08	13.59	13.33
$RRS_{n=500}$	47.36	86.55	**92.94**	**95.59**	**17.20**	13.75	13.46	13.27
$RRS_{n=1000}$	56.09	87.46	**93.24**	**95.85**	**16.25**	13.64	13.43	13.26
$RRS_{n=10000}$	61.43	87.68	**93.29**	**95.87**	**15.45**	13.59	13.40	13.25
PseudoDoc	70.52	92.90	**96.30**	**97.97**	**15.08**	13.53	13.43	13.38
	Prec@10				**rPrec**			
$RRS_{n=10}$	**26.80**	24.24	24.24	24.24	5.33	14.86	**17.78**	**18.74**
$RRS_{n=20}$	**25.42**	**24.24**	**24.24**	**24.24**	7.65	**18.04**	**19.32**	**19.74**
$RRS_{n=50}$	**26.55**	**25.97**	**25.97**	**25.97**	11.88	**20.39**	**20.90**	**21.02**
$RRS_{n=100}$	**27.99**	**27.41**	**27.41**	**27.41**	15.81	**21.85**	**22.16**	**22.28**
$RRS_{n=200}$	**28.63**	**28.13**	**28.13**	**28.13**	18.77	**22.25**	**22.56**	**22.68**
$RRS_{n=500}$	**29.28**	**28.71**	**28.71**	**28.71**	20.77	**23.11**	**23.42**	**23.54**
$RRS_{n=1000}$	**29.50**	**29.57**	**29.57**	**29.57**	21.02	**23.53**	**23.83**	**23.95**
$RRS_{n=10000}$	**30.43**	**30.00**	**30.00**	**30.00**	21.12	**23.65**	**23.95**	**24.07**
PseudoDoc	**26.47**	**26.62**	**26.62**	**26.62**	21.31	**22.75**	**22.86**	**22.92**
	AvgPrec				**AUC**			
$RRS_{n=10}$	2.78	7.00	10.24	**13.33**	3.77	8.03	11.43	**14.61**
$RRS_{n=20}$	4.16	9.62	13.32	**16.07**	5.00	10.77	14.63	**17.55**
$RRS_{n=50}$	6.34	14.10	17.42	**19.05**	7.22	15.55	19.01	**20.63**
$RRS_{n=100}$	8.52	17.14	19.59	**20.52**	9.61	18.66	21.28	**21.99**
$RRS_{n=200}$	10.85	19.16	20.79	**21.40**	12.13	20.77	22.35	**22.85**
$RRS_{n=500}$	14.15	20.78	**21.93**	**22.39**	15.51	22.22	**23.39**	**23.86**
$RRS_{n=1000}$	15.93	21.39	**22.46**	**22.92**	17.35	22.76	**23.93**	**24.40**
$RRS_{n=10000}$	16.96	21.47	**22.52**	**22.97**	18.27	22.78	**23.92**	**24.37**
PseudoDoc	17.57	21.72	**22.33**	**22.56**	19.21	23.36	**23.98**	**24.24**

Table A.24: Impact of k on PAR with α=100 (exalead, CAL500)

PAR, $k = 25$	RRS	α=1	α=5	α=10	α=20	α=40	α=50	α=100
	Prec@10							
$RRS_{n=10}$	**31.19**	30.45	**30.80**	**30.85**	**30.85**	**30.85**	**30.85**	**30.85**
$RRS_{n=20}$	**32.40**	31.05	**32.05**	**32.15**	**32.25**	**32.35**	**32.35**	**32.40**
$RRS_{n=50}$	**38.45**	34.35	**38.30**	**38.20**	**38.30**	**38.30**	**38.45**	**38.45**
$RRS_{n=100}$	**44.10**	33.85	**42.60**	**44.15**	**44.35**	**44.15**	**44.05**	**44.05**
$RRS_{n=200}$	**47.75**	31.90	**44.95**	**46.35**	**47.35**	**47.45**	**47.35**	**47.75**
$RRS_{n=500}$	**50.30**	24.40	43.90	**47.45**	**49.05**	**49.05**	**49.60**	**50.35**
$RRS_{n=1000}$	**52.55**	19.00	37.50	43.00	**47.75**	**50.65**	**51.70**	**52.30**
$RRS_{n=10000}$	**57.45**	9.85	14.20	17.50	22.55	29.75	32.45	**41.25**
$PseudoDoc$	**39.25**	5.80	7.00	8.55	11.10	15.20	17.40	**23.40**
	rPrec							
$RRS_{n=10}$	2.16	**5.11**	**5.15**	**5.15**	**5.15**	**5.15**	**5.15**	**5.15**
$RRS_{n=20}$	3.63	**7.39**	**7.48**	**7.48**	**7.48**	**7.48**	**7.48**	**7.48**
$RRS_{n=50}$	6.52	**10.77**	**11.13**	**11.15**	**11.15**	**11.15**	**11.15**	**11.15**
$RRS_{n=100}$	10.24	14.08	**14.64**	**14.64**	**14.64**	**14.64**	**14.64**	**14.63**
$RRS_{n=200}$	14.22	16.69	**17.81**	**17.85**	**17.86**	**17.84**	**17.84**	**17.83**
$RRS_{n=500}$	19.84	19.43	**21.93**	**22.04**	**22.06**	**22.04**	**22.04**	**22.04**
$RRS_{n=1000}$	24.22	18.51	**24.22**	**24.87**	**25.15**	**25.19**	**25.21**	**25.24**
$RRS_{n=10000}$	**35.20**	8.60	17.47	22.41	26.70	30.03	30.90	**33.12**
$PseudoDoc$	**30.78**	4.95	8.95	12.57	17.22	21.69	22.94	**26.22**
	AvgPrec							
$RRS_{n=10}$	1.19	1.76	**1.89**	**1.90**	**1.90**	**1.90**	**1.90**	**1.90**
$RRS_{n=20}$	1.84	2.76	**2.95**	**2.95**	**2.95**	**2.95**	**2.95**	**2.95**
$RRS_{n=50}$	3.24	4.60	**4.97**	**4.99**	**4.98**	**4.98**	**4.99**	**4.99**
$RRS_{n=100}$	5.54	7.00	**7.78**	**7.83**	**7.84**	**7.85**	**7.85**	**7.85**
$RRS_{n=200}$	8.23	9.25	10.75	**10.88**	**10.94**	**10.94**	**10.93**	**10.94**
$RRS_{n=500}$	12.39	11.40	14.65	15.00	**15.18**	**15.34**	**15.37**	**15.44**
$RRS_{n=1000}$	16.10	11.81	16.79	17.70	**18.28**	**18.64**	**18.72**	**18.88**
$RRS_{n=10000}$	29.98	8.37	14.69	17.95	21.16	24.07	**24.95**	**27.27**
$PseudoDoc$	**25.97**	5.47	9.13	11.71	14.75	17.85	18.79	**21.40**
	AUC							
$RRS_{n=10}$	3.05	3.30	**3.48**	**3.48**	**3.48**	**3.48**	**3.48**	**3.48**
$RRS_{n=20}$	3.64	4.16	**4.49**	**4.48**	**4.50**	**4.50**	**4.50**	**4.50**
$RRS_{n=50}$	4.99	5.95	**6.50**	**6.55**	**6.54**	**6.58**	**6.58**	**6.57**
$RRS_{n=100}$	7.11	8.33	9.30	9.37	**9.38**	**9.41**	**9.40**	**9.43**
$RRS_{n=200}$	9.62	10.48	12.10	12.25	**12.33**	**12.32**	**12.33**	**12.34**
$RRS_{n=500}$	13.76	12.71	16.02	16.38	**16.59**	**16.76**	**16.81**	**16.89**
$RRS_{n=1000}$	17.22	13.27	18.05	18.94	19.60	**19.88**	**19.95**	**20.11**
$RRS_{n=10000}$	**31.25**	9.43	16.21	19.53	22.79	25.67	26.45	**28.63**
$PseudoDoc$	**27.09**	6.27	10.25	12.96	16.00	19.13	20.12	**22.65**

Table A.25: Impact of α on PAR with $k = 25$ (Google, c35k)

PAR, $k = 50$	RRS	α=1	α=5	α=10	α=20	α=40	α=50	α=100
	Prec@10							
$RRS_{n=10}$	**31.19**	30.45	**30.80**	**30.85**	**30.85**	**30.85**	**30.85**	**30.85**
$RRS_{n=20}$	**32.40**	31.05	**32.05**	**32.15**	**32.25**	**32.35**	**32.35**	**32.40**
$RRS_{n=50}$	**38.45**	34.35	**38.30**	**38.20**	**38.30**	**38.30**	**38.45**	**38.45**
$RRS_{n=100}$	**44.10**	33.85	**42.60**	**44.15**	**44.35**	**44.15**	**44.05**	**44.05**
$RRS_{n=200}$	**47.75**	31.90	**44.95**	**46.35**	**47.35**	**47.45**	**47.35**	**47.75**
$RRS_{n=500}$	**50.30**	24.40	43.90	**47.45**	**49.05**	**49.05**	**49.60**	**50.35**
$RRS_{n=1000}$	**52.55**	19.00	37.50	43.00	**47.75**	**50.65**	**51.70**	**52.30**
$RRS_{n=10000}$	**57.45**	9.85	14.20	17.50	22.55	29.75	32.45	**41.25**
PseudoDoc	**39.25**	5.80	7.00	8.55	11.10	15.20	17.40	**23.40**
	rPrec							
$RRS_{n=10}$	2.16	**6.19**	**6.22**	**6.22**	**6.22**	**6.22**	**6.22**	**6.22**
$RRS_{n=20}$	3.63	**8.37**	**8.46**	**8.46**	**8.46**	**8.46**	**8.46**	**8.46**
$RRS_{n=50}$	6.52	**11.30**	**11.66**	**11.67**	**11.67**	**11.67**	**11.67**	**11.67**
$RRS_{n=100}$	10.24	14.19	**14.75**	**14.75**	**14.74**	**14.75**	**14.75**	**14.74**
$RRS_{n=200}$	14.22	16.70	**17.82**	**17.86**	**17.87**	**17.86**	**17.86**	**17.84**
$RRS_{n=500}$	19.84	19.43	**21.93**	**22.04**	**22.06**	**22.04**	**22.04**	**22.04**
$RRS_{n=1000}$	24.22	18.51	**24.22**	**24.87**	**25.15**	**25.19**	**25.21**	**25.24**
$RRS_{n=10000}$	**35.20**	8.60	17.47	22.41	26.70	30.03	30.90	**33.12**
PseudoDoc	**30.78**	4.95	8.95	12.57	17.22	21.69	22.94	**26.22**
	AvgPrec							
$RRS_{n=10}$	1.19	2.07	**2.20**	**2.21**	**2.21**	**2.21**	**2.21**	**2.21**
$RRS_{n=20}$	1.84	3.18	**3.37**	**3.38**	**3.38**	**3.38**	**3.38**	**3.38**
$RRS_{n=50}$	3.24	5.15	**5.52**	**5.54**	**5.54**	**5.54**	**5.54**	**5.54**
$RRS_{n=100}$	5.54	7.59	**8.37**	**8.42**	**8.43**	**8.44**	**8.44**	**8.45**
$RRS_{n=200}$	8.23	9.84	11.34	**11.47**	**11.53**	**11.52**	**11.52**	**11.53**
$RRS_{n=500}$	12.39	11.91	15.16	**15.51**	**15.69**	**15.85**	**15.88**	**15.95**
$RRS_{n=1000}$	16.10	12.21	17.19	18.10	**18.68**	**19.04**	**19.12**	**19.28**
$RRS_{n=10000}$	29.98	8.43	14.76	18.02	21.22	24.14	25.01	**27.34**
PseudoDoc	**25.97**	5.48	9.14	11.72	14.76	17.86	18.80	**21.41**
	AUC							
$RRS_{n=10}$	3.05	3.59	**3.77**	**3.77**	**3.77**	**3.77**	**3.77**	**3.77**
$RRS_{n=20}$	3.64	4.58	**4.90**	**4.90**	**4.91**	**4.91**	**4.91**	**4.91**
$RRS_{n=50}$	4.99	6.53	**7.09**	**7.13**	**7.13**	**7.16**	**7.17**	**7.15**
$RRS_{n=100}$	7.11	8.89	9.86	**9.93**	**9.94**	**9.97**	**9.96**	**10.00**
$RRS_{n=200}$	9.62	11.06	12.68	12.83	**12.91**	**12.91**	**12.91**	**12.93**
$RRS_{n=500}$	13.76	13.25	16.55	16.92	**17.13**	**17.29**	**17.35**	**17.42**
$RRS_{n=1000}$	17.22	13.69	18.47	19.36	20.01	**20.29**	**20.37**	**20.52**
$RRS_{n=10000}$	**31.25**	9.46	16.23	19.55	22.81	25.69	26.47	**28.66**
PseudoDoc	**27.09**	6.29	10.27	12.98	16.02	19.15	20.14	**22.67**

Table A.26: Impact of α on PAR with $k = 50$ (Google, c35k)

PAR, $k=25$	RRS	α=1	α=5	α=10	α=20	α=40	α=50	α=100
				Prec@10				
$RRS_{n=10}$	**27.40**	26.80	26.95	26.95	26.95	26.95	26.95	26.95
$RRS_{n=20}$	**30.30**	**29.75**	**30.30**	**30.35**	**30.35**	**30.35**	**30.35**	**30.35**
$RRS_{n=50}$	**32.35**	30.20	**32.25**	**32.25**	**32.35**	**32.30**	**32.30**	**32.35**
$RRS_{n=100}$	**35.90**	30.95	**35.05**	**35.55**	**35.80**	**35.75**	**35.80**	**35.85**
$RRS_{n=200}$	**33.80**	28.55	**35.20**	**34.65**	**34.80**	**34.25**	**34.15**	**33.95**
$RRS_{n=500}$	**34.60**	23.10	**37.00**	**36.00**	**35.90**	**35.90**	**35.60**	**34.95**
$RRS_{n=1000}$	**36.80**	19.20	**34.50**	**36.25**	**37.85**	**37.45**	**37.55**	**37.25**
$RRS_{n=10000}$	**42.55**	11.85	17.60	23.00	28.15	**34.10**	**35.65**	**38.45**
$PseudoDoc$	**34.95**	7.25	10.20	12.00	15.15	19.55	20.95	**26.65**
				rPrec				
$RRS_{n=10}$	1.42	**3.94**	**3.94**	**3.94**	**3.94**	**3.94**	**3.94**	**3.94**
$RRS_{n=20}$	2.48	**5.84**	**5.89**	**5.89**	**5.89**	**5.89**	**5.89**	**5.89**
$RRS_{n=50}$	4.73	**9.06**	**9.24**	**9.25**	**9.25**	**9.25**	**9.25**	**9.25**
$RRS_{n=100}$	7.41	11.70	**11.96**	**11.97**	**11.97**	**11.97**	**11.97**	**11.97**
$RRS_{n=200}$	10.76	14.23	**14.81**	**14.83**	**14.84**	**14.84**	**14.84**	**14.84**
$RRS_{n=500}$	15.70	17.16	**18.49**	**18.50**	**18.51**	**18.52**	**18.52**	**18.50**
$RRS_{n=1000}$	20.36	18.54	**21.83**	**21.85**	**21.95**	**21.98**	**21.98**	**21.99**
$RRS_{n=10000}$	**30.25**	10.69	21.04	24.40	26.88	**28.45**	**28.77**	**29.55**
$PseudoDoc$	**26.72**	6.68	11.76	15.26	19.19	22.43	23.21	**24.86**
				AvgPrec				
$RRS_{n=10}$	0.93	1.42	**1.53**	**1.53**	**1.53**	**1.53**	**1.53**	**1.53**
$RRS_{n=20}$	1.38	2.09	**2.23**	**2.23**	**2.23**	**2.23**	**2.23**	**2.23**
$RRS_{n=50}$	2.42	3.59	**3.84**	**3.85**	**3.85**	**3.84**	**3.84**	**3.84**
$RRS_{n=100}$	3.72	5.27	**5.66**	**5.67**	**5.69**	**5.69**	**5.69**	**5.69**
$RRS_{n=200}$	5.34	7.04	**7.80**	**7.83**	**7.84**	**7.85**	**7.86**	**7.86**
$RRS_{n=500}$	8.42	9.38	**11.17**	**11.27**	**11.34**	**11.36**	**11.37**	**11.39**
$RRS_{n=1000}$	11.96	11.08	**14.35**	**14.65**	**14.84**	**14.94**	**14.96**	**14.99**
$RRS_{n=10000}$	23.97	9.55	16.44	18.96	21.02	22.63	**23.06**	**24.04**
$PseudoDoc$	**21.16**	6.43	10.49	12.74	15.09	17.27	17.89	**19.46**
				AUC				
$RRS_{n=10}$	2.79	2.90	**3.10**	**3.12**	**3.12**	**3.12**	**3.12**	**3.12**
$RRS_{n=20}$	3.13	3.52	**3.74**	**3.74**	**3.75**	**3.75**	**3.75**	**3.75**
$RRS_{n=50}$	4.13	4.86	**5.24**	**5.23**	**5.24**	**5.24**	**5.24**	**5.24**
$RRS_{n=100}$	5.17	6.46	**7.02**	**7.03**	**7.04**	**7.05**	**7.05**	**7.08**
$RRS_{n=200}$	6.62	8.13	9.06	**9.09**	**9.09**	**9.12**	**9.12**	**9.14**
$RRS_{n=500}$	9.79	10.52	12.50	12.54	12.65	**12.73**	**12.72**	**12.70**
$RRS_{n=1000}$	13.36	12.34	15.79	16.04	16.16	**16.29**	**16.34**	**16.44**
$RRS_{n=10000}$	25.32	10.79	17.91	20.50	22.43	24.00	**24.47**	**25.39**
$PseudoDoc$	**22.29**	7.37	11.67	13.93	16.24	18.41	19.01	**20.60**

Table A.27: Impact of α on PAR with $k=25$ (exalead, c35k)

PAR, $k=50$	RRS	α=1	α=5	α=10	α=20	α=40	α=50	α=100
	Prec@10							
$RRS_{n=10}$	**27.40**	26.80	26.95	26.95	26.95	26.95	26.95	26.95
$RRS_{n=20}$	**30.30**	**29.75**	**30.30**	**30.35**	**30.35**	**30.35**	**30.35**	**30.35**
$RRS_{n=50}$	**32.35**	30.20	**32.25**	**32.25**	**32.35**	**32.30**	**32.30**	**32.35**
$RRS_{n=100}$	**35.90**	30.95	**35.05**	**35.55**	**35.80**	**35.75**	**35.80**	**35.85**
$RRS_{n=200}$	**33.80**	28.55	**35.20**	**34.65**	**34.80**	**34.25**	**34.15**	**33.95**
$RRS_{n=500}$	**34.60**	23.10	**37.00**	**36.00**	**35.90**	**35.90**	**35.60**	**34.95**
$RRS_{n=1000}$	**36.80**	19.20	**34.50**	**36.25**	**37.85**	**37.45**	**37.55**	**37.25**
$RRS_{n=10000}$	**42.55**	11.85	17.60	23.00	28.15	**34.10**	**35.65**	**38.45**
$PseudoDoc$	**34.95**	7.25	10.20	12.00	15.15	19.55	20.95	**26.65**
	rPrec							
$RRS_{n=10}$	1.42	**4.94**	**4.93**	**4.94**	**4.94**	**4.94**	**4.94**	**4.94**
$RRS_{n=20}$	2.48	**6.87**	**6.92**	**6.92**	**6.92**	**6.92**	**6.92**	**6.92**
$RRS_{n=50}$	4.73	**9.70**	**9.88**	**9.89**	**9.89**	**9.89**	**9.89**	**9.89**
$RRS_{n=100}$	7.41	11.95	**12.21**	**12.21**	**12.21**	**12.21**	**12.21**	**12.21**
$RRS_{n=200}$	10.76	14.31	**14.89**	**14.90**	**14.92**	**14.92**	**14.92**	**14.92**
$RRS_{n=500}$	15.70	17.18	**18.51**	**18.52**	**18.52**	**18.53**	**18.53**	**18.52**
$RRS_{n=1000}$	20.36	18.54	**21.83**	**21.85**	**21.95**	**21.98**	**21.98**	**21.99**
$RRS_{n=10000}$	**30.25**	10.69	21.04	24.40	26.88	**28.45**	**28.77**	**29.55**
$PseudoDoc$	**26.72**	6.68	11.76	15.26	19.19	22.43	23.21	**24.86**
	AvgPrec							
$RRS_{n=10}$	0.93	1.67	**1.79**	**1.79**	**1.79**	**1.79**	**1.79**	**1.79**
$RRS_{n=20}$	1.38	2.43	**2.57**	**2.57**	**2.57**	**2.57**	**2.57**	**2.57**
$RRS_{n=50}$	2.42	4.08	**4.34**	**4.34**	**4.34**	**4.34**	**4.34**	**4.34**
$RRS_{n=100}$	3.72	5.85	**6.24**	**6.25**	**6.27**	**6.27**	**6.27**	**6.27**
$RRS_{n=200}$	5.34	7.65	**8.41**	**8.44**	**8.45**	**8.46**	**8.47**	**8.47**
$RRS_{n=500}$	8.42	9.94	**11.73**	**11.83**	**11.90**	**11.92**	**11.94**	**11.95**
$RRS_{n=1000}$	11.96	11.54	14.82	**15.12**	**15.31**	**15.41**	**15.43**	**15.45**
$RRS_{n=10000}$	23.97	9.65	16.54	19.06	21.12	22.73	**23.16**	**24.14**
$PseudoDoc$	**21.16**	6.45	10.52	12.77	15.12	17.29	17.92	**19.49**
	AUC							
$RRS_{n=10}$	2.79	3.13	**3.34**	**3.36**	**3.36**	**3.36**	**3.36**	**3.36**
$RRS_{n=20}$	3.13	3.84	**4.06**	**4.06**	**4.06**	**4.06**	**4.06**	**4.06**
$RRS_{n=50}$	4.13	5.37	**5.75**	**5.74**	**5.75**	**5.75**	**5.75**	**5.75**
$RRS_{n=100}$	5.17	7.08	**7.64**	**7.65**	**7.66**	**7.67**	**7.67**	**7.70**
$RRS_{n=200}$	6.62	8.75	9.67	**9.71**	**9.71**	**9.74**	**9.74**	**9.76**
$RRS_{n=500}$	9.79	11.07	13.05	13.09	13.20	**13.28**	**13.27**	**13.25**
$RRS_{n=1000}$	13.36	12.79	16.24	16.50	16.61	**16.74**	**16.80**	**16.89**
$RRS_{n=10000}$	25.32	10.86	17.99	20.58	22.50	24.07	**24.54**	**25.46**
$PseudoDoc$	**22.29**	7.39	11.69	13.95	16.27	18.43	19.03	**20.62**

Table A.28: Impact of α on PAR with $k=50$ (exalead, c35k)

PAR, $k = 25$	RRS	α=1	α=5	α=10	α=20	α=40	α=50	α=100
	Prec@10							
$RRS_{n=10}$	**25.77**	22.16	23.88	23.74	23.74	23.74	23.74	23.74
$RRS_{n=20}$	**25.98**	24.24	**25.61**	**25.61**	**25.61**	**25.61**	**25.61**	**25.61**
$RRS_{n=50}$	**26.06**	**26.69**	**26.12**	**26.04**	**26.04**	**26.04**	**26.04**	**26.04**
$RRS_{n=100}$	**29.30**	**28.35**	**29.35**	**29.50**	**29.28**	**29.28**	**29.28**	**29.28**
$RRS_{n=200}$	**30.60**	**28.78**	**31.29**	**30.94**	**30.86**	**31.08**	**30.79**	**30.58**
$RRS_{n=500}$	**32.68**	25.83	**31.37**	**33.02**	**33.09**	**33.02**	**33.17**	**32.88**
$RRS_{n=1000}$	**33.47**	25.47	29.71	**31.44**	**33.45**	**33.81**	**33.60**	**33.74**
$RRS_{n=10000}$	**33.62**	23.31	28.99	**30.29**	**32.66**	**32.88**	**33.67**	**33.38**
PseudoDoc	**30.72**	23.53	**27.34**	**27.63**	**29.64**	**31.08**	**31.15**	**31.01**
	rPrec							
$RRS_{n=10}$	5.61	**17.16**	**17.30**	**17.39**	**17.39**	**17.39**	**17.39**	**17.39**
$RRS_{n=20}$	8.84	19.39	**20.32**	**20.35**	**20.35**	**20.35**	**20.35**	**20.35**
$RRS_{n=50}$	13.49	21.74	**22.02**	**22.20**	**22.22**	**22.22**	**22.22**	**22.22**
$RRS_{n=100}$	18.05	**22.65**	**23.69**	**23.67**	**23.71**	**23.76**	**23.78**	**23.78**
$RRS_{n=200}$	21.58	**23.06**	**24.55**	**24.38**	**24.30**	**24.41**	**24.37**	**24.38**
$RRS_{n=500}$	**24.06**	22.28	**24.30**	**25.82**	**25.75**	**25.74**	**25.90**	**25.88**
$RRS_{n=1000}$	**24.86**	20.78	**23.77**	**24.89**	**26.27**	**26.59**	**26.51**	**26.46**
$RRS_{n=10000}$	**25.06**	19.42	22.79	**24.43**	**25.71**	**26.69**	**26.76**	**26.81**
PseudoDoc	**25.77**	18.92	**22.96**	**24.15**	**25.67**	**26.22**	**26.28**	**26.64**
	AvgPrec							
$RRS_{n=10}$	3.58	**10.29**	**10.46**	**10.46**	**10.46**	**10.46**	**10.46**	**10.46**
$RRS_{n=20}$	5.30	**14.01**	**14.35**	**14.35**	**14.35**	**14.35**	**14.35**	**14.35**
$RRS_{n=50}$	7.57	18.11	**18.59**	**18.58**	**18.61**	**18.61**	**18.61**	**18.61**
$RRS_{n=100}$	10.59	20.88	**21.64**	**21.64**	**21.64**	**21.62**	**21.61**	**21.61**
$RRS_{n=200}$	13.84	**22.02**	**23.58**	**23.68**	**23.59**	**23.48**	**23.49**	**23.48**
$RRS_{n=500}$	18.02	21.56	**23.86**	**24.58**	**24.81**	**24.97**	**25.01**	**24.99**
$RRS_{n=1000}$	20.37	20.48	**23.38**	**24.24**	**24.86**	**25.33**	**25.39**	**25.67**
$RRS_{n=10000}$	21.77	19.09	**22.42**	**23.66**	**24.78**	**25.44**	**25.57**	**25.93**
PseudoDoc	22.66	19.45	**22.56**	**23.77**	**24.91**	**25.33**	**25.63**	**25.78**
	AUC							
$RRS_{n=10}$	4.50	**11.35**	**11.56**	**11.56**	**11.56**	**11.56**	**11.56**	**11.56**
$RRS_{n=20}$	6.13	**15.34**	**15.72**	**15.72**	**15.72**	**15.72**	**15.72**	**15.72**
$RRS_{n=50}$	8.63	19.61	**20.16**	**20.12**	**20.16**	**20.16**	**20.16**	**20.16**
$RRS_{n=100}$	11.69	**22.55**	**23.21**	**23.26**	**23.25**	**23.23**	**23.23**	**23.23**
$RRS_{n=200}$	15.16	23.41	**25.05**	**25.12**	**25.00**	**24.96**	**24.97**	**24.95**
$RRS_{n=500}$	19.33	22.93	**25.24**	**25.91**	**26.10**	**26.31**	**26.39**	**26.41**
$RRS_{n=1000}$	21.77	21.69	**24.79**	**25.57**	**26.21**	**26.75**	**26.77**	**27.07**
$RRS_{n=10000}$	23.16	20.28	**23.88**	**25.14**	**26.26**	**26.89**	**27.03**	**27.35**
PseudoDoc	24.12	20.47	**23.92**	**25.20**	**26.42**	**26.78**	**27.10**	**27.25**

Table A.29: Impact of α on PAR with $k = 25$ (Google, CAL500)

PAR, $k = 50$	RRS	α=1	α=5	α=10	α=20	α=40	α=50	α=100
	Prec@10							
$RRS_{n=10}$	**25.77**	22.16	23.88	23.74	23.74	23.74	23.74	23.74
$RRS_{n=20}$	**25.98**	24.24	**25.61**	**25.61**	**25.61**	**25.61**	**25.61**	**25.61**
$RRS_{n=50}$	**26.06**	**26.69**	**26.12**	**26.04**	**26.04**	**26.04**	**26.04**	**26.04**
$RRS_{n=100}$	**29.30**	**28.35**	**29.35**	**29.50**	**29.28**	**29.28**	**29.28**	**29.28**
$RRS_{n=200}$	**30.60**	**28.78**	**31.29**	**30.94**	**30.86**	**31.08**	**30.79**	**30.58**
$RRS_{n=500}$	**32.68**	25.83	**31.37**	**33.02**	**33.09**	**33.02**	**33.17**	**32.88**
$RRS_{n=1000}$	**33.47**	25.47	29.71	**31.44**	**33.45**	**33.81**	**33.60**	**33.74**
$RRS_{n=10000}$	**33.62**	23.31	28.99	**30.29**	**32.66**	**32.88**	**33.67**	**33.38**
$PseudoDoc$	**30.72**	23.53	**27.34**	**27.63**	**29.64**	**31.08**	**31.15**	**31.01**
	rPrec							
$RRS_{n=10}$	5.61	**18.21**	**18.35**	**18.44**	**18.44**	**18.44**	**18.44**	**18.44**
$RRS_{n=20}$	8.84	19.74	**20.67**	**20.70**	**20.70**	**20.70**	**20.70**	**20.70**
$RRS_{n=50}$	13.49	21.74	**22.03**	**22.20**	**22.23**	**22.23**	**22.23**	**22.23**
$RRS_{n=100}$	18.05	**22.66**	**23.70**	**23.67**	**23.72**	**23.77**	**23.79**	**23.79**
$RRS_{n=200}$	21.58	**23.07**	**24.56**	**24.39**	**24.31**	**24.41**	**24.38**	**24.38**
$RRS_{n=500}$	**24.06**	22.28	**24.30**	**25.83**	**25.75**	**25.74**	**25.91**	**25.89**
$RRS_{n=1000}$	**24.86**	20.78	**23.77**	**24.89**	**26.27**	**26.59**	**26.52**	**26.46**
$RRS_{n=10000}$	**25.06**	19.43	22.79	**24.44**	**25.71**	**26.70**	**26.76**	**26.81**
$PseudoDoc$	**25.77**	18.92	**22.96**	**24.15**	**25.67**	**26.22**	**26.28**	**26.64**
	AvgPrec							
$RRS_{n=10}$	3.58	**13.34**	**13.51**	**13.51**	**13.51**	**13.51**	**13.51**	**13.51**
$RRS_{n=20}$	5.30	**16.86**	**17.20**	**17.20**	**17.20**	**17.20**	**17.20**	**17.20**
$RRS_{n=50}$	7.57	19.69	**20.17**	**20.16**	**20.19**	**20.19**	**20.19**	**20.19**
$RRS_{n=100}$	10.59	21.68	**22.44**	**22.44**	**22.44**	**22.41**	**22.41**	**22.41**
$RRS_{n=200}$	13.84	**22.43**	**23.99**	**24.09**	**24.00**	**23.90**	**23.90**	**23.90**
$RRS_{n=500}$	18.02	21.82	**24.12**	**24.84**	**25.07**	**25.24**	**25.27**	**25.25**
$RRS_{n=1000}$	20.37	20.73	**23.63**	**24.49**	**25.11**	**25.58**	**25.64**	**25.92**
$RRS_{n=10000}$	21.77	19.34	**22.67**	**23.91**	**25.03**	**25.68**	**25.82**	**26.18**
$PseudoDoc$	22.66	19.56	**22.68**	**23.89**	**25.02**	**25.45**	**25.74**	**25.89**
	AUC							
$RRS_{n=10}$	4.50	**14.56**	**14.77**	**14.77**	**14.77**	**14.77**	**14.77**	**14.77**
$RRS_{n=20}$	6.13	**18.26**	**18.64**	**18.64**	**18.64**	**18.64**	**18.64**	**18.64**
$RRS_{n=50}$	8.63	21.14	**21.69**	**21.65**	**21.70**	**21.69**	**21.69**	**21.69**
$RRS_{n=100}$	11.69	**23.02**	**23.68**	**23.73**	**23.72**	**23.70**	**23.70**	**23.70**
$RRS_{n=200}$	15.16	23.73	**25.37**	**25.44**	**25.32**	**25.28**	**25.29**	**25.28**
$RRS_{n=500}$	19.33	23.20	**25.52**	**26.18**	**26.37**	**26.58**	**26.66**	**26.68**
$RRS_{n=1000}$	21.77	21.95	**25.06**	**25.84**	**26.48**	**27.02**	**27.04**	**27.34**
$RRS_{n=10000}$	23.16	20.52	**24.12**	**25.39**	**26.50**	**27.13**	**27.27**	**27.59**
$PseudoDoc$	24.12	20.63	**24.08**	**25.36**	**26.58**	**26.95**	**27.26**	**27.41**

Table A.30: Impact of α on PAR with $k = 50$ (Google, CAL500)

PAR, $k = 25$	RRS	α=1	α=5	α=10	α=20	α=40	α=50	α=100
	Prec@10							
$RRS_{n=10}$	**26.80**	22.73	24.17	24.24	24.24	24.24	24.24	24.24
$RRS_{n=20}$	**25.42**	**23.88**	**24.24**	**24.24**	**24.24**	**24.24**	**24.24**	**24.24**
$RRS_{n=50}$	**26.55**	**26.19**	**26.12**	**26.04**	**25.97**	**25.97**	**25.97**	**25.97**
$RRS_{n=100}$	**27.99**	**27.12**	**27.12**	**27.27**	**27.41**	**27.48**	**27.41**	**27.41**
$RRS_{n=200}$	**28.63**	**28.06**	**28.06**	**27.84**	**28.20**	**28.06**	**27.99**	**28.13**
$RRS_{n=500}$	**29.28**	**26.91**	**28.13**	**29.21**	**28.71**	**28.71**	**28.78**	**28.71**
$RRS_{n=1000}$	**29.50**	24.82	**28.20**	**28.99**	**29.50**	**29.86**	**30.00**	**29.57**
$RRS_{n=10000}$	**30.43**	23.38	**26.69**	**27.77**	**28.71**	**30.14**	**30.07**	**30.00**
$PseudoDoc$	**26.47**	**23.88**	**26.04**	**26.33**	**27.27**	**27.41**	**27.27**	**26.62**
	rPrec							
$RRS_{n=10}$	5.33	**17.38**	**17.78**	**17.78**	**17.78**	**17.78**	**17.78**	**17.78**
$RRS_{n=20}$	7.65	**18.94**	**19.32**	**19.32**	**19.32**	**19.32**	**19.32**	**19.32**
$RRS_{n=50}$	11.88	**20.80**	**20.85**	**20.88**	**20.90**	**20.90**	**20.90**	**20.90**
$RRS_{n=100}$	15.81	**21.19**	**22.33**	**22.18**	**22.16**	**22.17**	**22.17**	**22.16**
$RRS_{n=200}$	18.77	**21.99**	**22.84**	**22.67**	**22.70**	**22.55**	**22.54**	**22.56**
$RRS_{n=500}$	20.77	**21.10**	**22.87**	**23.39**	**23.62**	**23.59**	**23.46**	**23.42**
$RRS_{n=1000}$	21.02	**20.11**	**22.56**	**23.20**	**23.77**	**23.87**	**23.83**	**23.83**
$RRS_{n=10000}$	**21.12**	18.77	**21.03**	**22.19**	**23.21**	**23.42**	**23.65**	**23.95**
$PseudoDoc$	**21.31**	18.88	**21.28**	**22.41**	**22.72**	**23.00**	**22.92**	**22.86**
	AvgPrec							
$RRS_{n=10}$	2.78	**10.10**	**10.24**	**10.24**	**10.24**	**10.24**	**10.24**	**10.24**
$RRS_{n=20}$	4.16	**13.20**	**13.30**	**13.32**	**13.32**	**13.32**	**13.32**	**13.32**
$RRS_{n=50}$	6.34	**17.37**	**17.49**	**17.49**	**17.42**	**17.42**	**17.42**	**17.42**
$RRS_{n=100}$	8.52	**19.49**	**19.78**	**19.70**	**19.68**	**19.62**	**19.59**	**19.59**
$RRS_{n=200}$	10.85	**20.55**	**21.03**	**20.93**	**20.91**	**20.84**	**20.83**	20.79
$RRS_{n=500}$	14.15	**20.36**	**21.79**	**21.90**	**21.92**	**21.99**	**22.00**	**21.93**
$RRS_{n=1000}$	15.93	**19.46**	**21.51**	**22.01**	**22.32**	**22.42**	**22.45**	**22.46**
$RRS_{n=10000}$	16.96	**18.10**	**20.34**	**21.05**	**21.77**	**22.32**	**22.42**	**22.52**
$PseudoDoc$	17.57	18.77	**20.95**	**21.84**	**22.36**	**22.48**	**22.44**	**22.33**
	AUC							
$RRS_{n=10}$	3.77	**11.23**	**11.42**	**11.43**	**11.43**	**11.43**	**11.43**	**11.43**
$RRS_{n=20}$	5.00	**14.41**	**14.59**	**14.63**	**14.63**	**14.63**	**14.63**	**14.63**
$RRS_{n=50}$	7.22	**18.72**	**19.00**	**19.05**	**19.01**	**19.01**	**19.01**	**19.01**
$RRS_{n=100}$	9.61	**20.97**	**21.43**	**21.33**	**21.39**	**21.31**	**21.28**	**21.28**
$RRS_{n=200}$	12.13	**21.82**	**22.40**	**22.30**	**22.38**	**22.38**	**22.39**	**22.35**
$RRS_{n=500}$	15.51	**21.62**	**23.15**	**23.23**	**23.34**	**23.44**	**23.48**	**23.39**
$RRS_{n=1000}$	17.35	20.77	**22.85**	**23.36**	**23.73**	**23.85**	**23.92**	**23.93**
$RRS_{n=10000}$	18.27	19.36	**21.70**	**22.51**	**23.21**	**23.80**	**23.92**	**23.92**
$PseudoDoc$	19.21	20.07	**22.40**	**23.28**	**23.74**	**23.93**	**23.95**	**23.98**

Table A.31: Impact of α on PAR with $k = 25$ (exalead, CAL500)

PAR, $k = 50$	RRS	α=1	α=5	α=10	α=20	α=40	α=50	α=100
	Prec@10							
$RRS_{n=10}$	**26.80**	22.73	24.17	24.24	24.24	24.24	24.24	24.24
$RRS_{n=20}$	**25.42**	**23.88**	**24.24**	**24.24**	**24.24**	**24.24**	**24.24**	**24.24**
$RRS_{n=50}$	**26.55**	**26.19**	**26.12**	**26.04**	**25.97**	**25.97**	**25.97**	**25.97**
$RRS_{n=100}$	**27.99**	**27.12**	**27.12**	**27.27**	**27.41**	**27.48**	**27.41**	**27.41**
$RRS_{n=200}$	**28.63**	**28.06**	**28.06**	**27.84**	**28.20**	**28.06**	**27.99**	**28.13**
$RRS_{n=500}$	**29.28**	**26.91**	**28.13**	**29.21**	**28.71**	**28.71**	**28.78**	**28.71**
$RRS_{n=1000}$	**29.50**	24.82	**28.20**	**28.99**	**29.50**	**29.86**	**30.00**	**29.57**
$RRS_{n=10000}$	**30.43**	23.38	**26.69**	**27.77**	**28.71**	**30.14**	**30.07**	**30.00**
$PseudoDoc$	**26.47**	**23.88**	**26.04**	**26.33**	**27.27**	**27.41**	**27.27**	**26.62**
	rPrec							
$RRS_{n=10}$	5.33	**18.33**	**18.74**	**18.74**	**18.74**	**18.74**	**18.74**	**18.74**
$RRS_{n=20}$	7.65	**19.36**	**19.74**	**19.74**	**19.74**	**19.74**	**19.74**	**19.74**
$RRS_{n=50}$	11.88	**20.93**	**20.97**	**21.01**	**21.02**	**21.02**	**21.02**	**21.02**
$RRS_{n=100}$	15.81	**21.31**	**22.45**	**22.30**	**22.28**	**22.29**	**22.29**	**22.28**
$RRS_{n=200}$	18.77	**22.12**	**22.96**	**22.80**	**22.82**	**22.67**	**22.67**	**22.68**
$RRS_{n=500}$	20.77	**21.23**	**22.99**	**23.51**	**23.75**	**23.71**	**23.58**	**23.54**
$RRS_{n=1000}$	21.02	**20.23**	**22.68**	**23.33**	**23.89**	**24.00**	**23.96**	**23.95**
$RRS_{n=10000}$	**21.12**	18.90	**21.15**	**22.32**	**23.33**	**23.55**	**23.78**	**24.07**
$PseudoDoc$	**21.31**	18.93	**21.34**	**22.47**	**22.78**	**23.06**	**22.98**	**22.92**
	AvgPrec							
$RRS_{n=10}$	2.78	**13.18**	**13.33**	**13.33**	**13.33**	**13.33**	**13.33**	**13.33**
$RRS_{n=20}$	4.16	**15.94**	**16.05**	**16.06**	**16.07**	**16.07**	**16.07**	**16.07**
$RRS_{n=50}$	6.34	**19.00**	**19.12**	**19.12**	**19.05**	**19.05**	**19.05**	**19.05**
$RRS_{n=100}$	8.52	**20.43**	**20.72**	**20.63**	**20.61**	**20.55**	**20.53**	**20.52**
$RRS_{n=200}$	10.85	**21.15**	**21.64**	**21.54**	**21.51**	**21.44**	**21.44**	21.40
$RRS_{n=500}$	14.15	**20.82**	**22.25**	**22.36**	**22.38**	**22.46**	**22.47**	**22.39**
$RRS_{n=1000}$	15.93	**19.92**	**21.97**	**22.46**	**22.78**	**22.88**	**22.91**	**22.92**
$RRS_{n=10000}$	16.96	**18.55**	**20.79**	**21.50**	**22.22**	**22.77**	**22.87**	**22.97**
$PseudoDoc$	17.57	19.00	**21.18**	**22.07**	**22.59**	**22.71**	**22.67**	**22.56**
	AUC							
$RRS_{n=10}$	3.77	**14.41**	**14.61**	**14.61**	**14.61**	**14.61**	**14.61**	**14.61**
$RRS_{n=20}$	5.00	**17.33**	**17.51**	**17.55**	**17.55**	**17.55**	**17.55**	**17.55**
$RRS_{n=50}$	7.22	**20.33**	**20.61**	**20.66**	**20.63**	**20.63**	**20.63**	**20.63**
$RRS_{n=100}$	9.61	**21.68**	**22.14**	**22.04**	**22.10**	**22.02**	**21.99**	**21.99**
$RRS_{n=200}$	12.13	**22.32**	**22.89**	**22.80**	**22.88**	**22.87**	**22.88**	**22.85**
$RRS_{n=500}$	15.51	**22.09**	**23.62**	**23.70**	**23.80**	**23.90**	**23.94**	**23.86**
$RRS_{n=1000}$	17.35	21.23	**23.32**	**23.82**	**24.19**	**24.31**	**24.38**	**24.40**
$RRS_{n=10000}$	18.27	19.81	**22.15**	**22.96**	**23.66**	**24.25**	**24.37**	**24.37**
$PseudoDoc$	19.21	20.33	**22.66**	**23.54**	**24.00**	**24.19**	**24.21**	**24.24**

Table A.32: Impact of α on PAR with $k = 50$ (exalead, CAL500)

	RRS	+PAR, $k=50$		QC	+PAR, $k=50$	
		$\alpha=10$	$\alpha=100$		$\alpha=10$	$\alpha=100$
	Rec					
$RRS_{n=10}$	2.18	**10.71**	**10.71**	2.89	**10.89**	**10.89**
$RRS_{n=20}$	3.74	**16.89**	**16.89**	4.50	**16.24**	**16.24**
$RRS_{n=50}$	7.17	**27.76**	**27.76**	8.81	**26.69**	**26.69**
$RRS_{n=100}$	12.72	**39.41**	**39.41**	14.42	**38.17**	**38.17**
$RRS_{n=200}$	18.67	**50.98**	**50.98**	21.80	**51.99**	**51.99**
$RRS_{n=500}$	29.31	**66.60**	**66.60**	32.34	**67.24**	**67.24**
$RRS_{n=1000}$	40.38	**77.63**	**77.63**	43.60	**78.84**	**78.84**
$RRS_{n=10000}$	80.50	**96.68**	**96.68**	76.02	**94.13**	**94.13**
	Prec					
$RRS_{n=10}$	**30.15**	6.81	6.81	**34.97**	7.35	7.35
$RRS_{n=20}$	**29.02**	6.57	6.57	**33.78**	7.01	7.01
$RRS_{n=50}$	**27.61**	6.17	6.17	**33.32**	6.68	6.68
$RRS_{n=100}$	**25.99**	5.66	5.66	**30.65**	6.13	6.13
$RRS_{n=200}$	**23.77**	5.25	5.25	**28.39**	5.61	5.61
$RRS_{n=500}$	**20.12**	4.81	4.81	**23.32**	5.01	5.01
$RRS_{n=1000}$	**16.88**	4.50	4.50	**19.14**	4.66	4.66
$RRS_{n=10000}$	7.29	3.85	3.85	**9.13**	3.94	3.94
	Prec@10					
$RRS_{n=10}$	**31.19**	**30.85**	**30.85**	**35.72**	**34.70**	**34.75**
$RRS_{n=20}$	32.40	32.15	32.40	**37.38**	**37.00**	**37.10**
$RRS_{n=50}$	**38.45**	**38.20**	**38.45**	**40.05**	**39.85**	**40.00**
$RRS_{n=100}$	**44.10**	**44.15**	**44.05**	**45.00**	**45.30**	**45.15**
$RRS_{n=200}$	**47.75**	**46.35**	**47.75**	**49.20**	**49.00**	**48.90**
$RRS_{n=500}$	**50.30**	**47.45**	**50.35**	**51.05**	**50.00**	**51.30**
$RRS_{n=1000}$	**52.55**	43.00	**52.30**	**52.55**	46.05	**53.50**
$RRS_{n=10000}$	**57.45**	17.50	41.25	**61.25**	24.70	**48.90**
	rPrec					
$RRS_{n=10}$	2.16	**6.22**	**6.22**	2.67	**6.88**	**6.88**
$RRS_{n=20}$	3.63	**8.46**	**8.46**	4.24	**9.01**	**9.01**
$RRS_{n=50}$	6.52	11.67	11.67	8.47	**14.02**	**14.03**
$RRS_{n=100}$	10.24	14.75	14.74	12.43	**17.33**	**17.33**
$RRS_{n=200}$	14.22	17.86	17.84	18.13	**21.92**	**21.97**
$RRS_{n=500}$	19.84	22.04	22.04	23.72	**25.98**	**26.08**
$RRS_{n=1000}$	24.22	24.87	25.24	28.55	**28.91**	**29.67**
$RRS_{n=10000}$	35.20	22.41	33.12	**36.41**	27.71	**35.24**
	AvgPrec					
$RRS_{n=10}$	1.19	**2.21**	**2.21**	1.78	**2.95**	**2.94**
$RRS_{n=20}$	1.84	**3.38**	**3.38**	2.70	**4.29**	**4.30**
$RRS_{n=50}$	3.24	**5.54**	**5.54**	5.14	**7.66**	**7.67**
$RRS_{n=100}$	5.54	8.42	**8.45**	7.71	**10.84**	**10.85**
$RRS_{n=200}$	8.23	11.47	11.53	11.64	**15.12**	**15.18**
$RRS_{n=500}$	12.39	15.51	15.95	15.90	**18.99**	**19.54**
$RRS_{n=1000}$	16.10	18.10	19.28	20.56	22.31	**23.76**
$RRS_{n=10000}$	29.98	18.02	27.34	**31.64**	22.36	**30.58**

Table A.33: Impact of PAR on filtered RRS (Google, c35k)

	RRS	+PAR, $k=50$		QC	+PAR, $k=50$	
		$\alpha=10$	$\alpha=100$		$\alpha=10$	$\alpha=100$
	Rec					
$RRS_{n=10}$	1.73	**7.76**	**7.76**	1.49	**6.95**	**6.95**
$RRS_{n=20}$	2.80	**12.31**	**12.31**	3.03	**12.81**	**12.81**
$RRS_{n=50}$	5.46	**22.59**	**22.59**	6.22	**22.60**	**22.60**
$RRS_{n=100}$	8.71	**32.73**	**32.73**	10.30	**33.55**	**33.55**
$RRS_{n=200}$	13.40	**44.03**	**44.03**	17.56	**46.50**	**46.50**
$RRS_{n=500}$	23.19	60.14	60.14	28.55	**64.02**	**64.02**
$RRS_{n=1000}$	34.01	72.12	72.12	37.51	**73.73**	**73.73**
$RRS_{n=10000}$	72.48	**93.66**	**93.66**	67.27	89.56	89.56
	Prec					
$RRS_{n=10}$	**28.42**	6.64	6.64	**29.77**	6.87	6.87
$RRS_{n=20}$	**28.05**	6.81	6.81	**28.46**	6.50	6.50
$RRS_{n=50}$	**26.43**	6.20	6.20	**30.09**	6.34	6.34
$RRS_{n=100}$	**25.06**	5.79	5.79	**29.14**	6.09	6.09
$RRS_{n=200}$	**23.01**	5.42	5.42	**27.75**	5.66	5.66
$RRS_{n=500}$	**20.41**	4.99	4.99	**22.67**	5.02	5.02
$RRS_{n=1000}$	**17.97**	4.66	4.66	**20.04**	4.70	4.70
$RRS_{n=10000}$	**8.81**	3.91	3.91	**11.21**	4.03	4.03
	Prec@10					
$RRS_{n=10}$	**27.40**	**26.95**	**26.95**	**32.22**	**30.90**	**30.90**
$RRS_{n=20}$	**30.30**	**30.35**	**30.35**	**33.48**	**33.50**	**33.50**
$RRS_{n=50}$	32.35	32.25	32.35	**39.45**	**39.55**	**39.45**
$RRS_{n=100}$	35.90	35.55	35.85	**41.65**	**41.55**	**41.65**
$RRS_{n=200}$	33.80	34.65	33.95	**44.50**	**44.15**	**44.45**
$RRS_{n=500}$	34.60	36.00	34.95	**45.75**	**45.25**	**45.40**
$RRS_{n=1000}$	36.80	36.25	37.25	**48.60**	**45.85**	**48.10**
$RRS_{n=10000}$	42.55	23.00	38.45	**55.60**	31.40	**47.20**
	rPrec					
$RRS_{n=10}$	1.42	**4.94**	**4.94**	1.49	**4.84**	**4.85**
$RRS_{n=20}$	2.48	**6.92**	**6.92**	2.76	**7.10**	**7.10**
$RRS_{n=50}$	4.73	**9.89**	**9.89**	5.82	**11.10**	**11.10**
$RRS_{n=100}$	7.41	12.21	12.21	9.06	**14.22**	**14.23**
$RRS_{n=200}$	10.76	14.90	14.92	13.71	**17.98**	**17.98**
$RRS_{n=500}$	15.70	18.52	18.52	20.13	**22.71**	**22.70**
$RRS_{n=1000}$	20.36	21.85	21.99	25.01	**26.22**	**26.43**
$RRS_{n=10000}$	**30.25**	24.40	29.55	**31.66**	28.00	**31.32**
	AvgPrec					
$RRS_{n=10}$	0.93	**1.79**	**1.79**	0.78	**1.50**	**1.50**
$RRS_{n=20}$	1.38	**2.57**	**2.57**	1.59	**2.84**	**2.84**
$RRS_{n=50}$	2.42	**4.34**	**4.34**	3.18	**5.23**	**5.23**
$RRS_{n=100}$	3.72	6.25	6.27	4.98	**7.75**	**7.76**
$RRS_{n=200}$	5.34	8.44	8.47	7.88	**11.22**	**11.22**
$RRS_{n=500}$	8.42	11.83	11.95	12.21	**15.63**	**15.83**
$RRS_{n=1000}$	11.96	15.12	15.45	16.17	**19.16**	**19.65**
$RRS_{n=10000}$	23.97	19.06	24.14	25.55	22.38	**26.36**

Table A.34: Impact of PAR on filtered RRS (exalead, c35k)

	RRS	+PAR, $k=50$		QC	+PAR, $k=50$	
		$\alpha=10$	$\alpha=100$		$\alpha=10$	$\alpha=100$
	Rec					
$RRS_{n=10}$	5.96	**59.08**	**59.08**	5.95	**57.95**	**57.95**
$RRS_{n=20}$	10.19	**75.90**	**75.90**	9.04	**70.59**	**70.59**
$RRS_{n=50}$	17.99	**89.52**	**89.52**	17.93	**84.10**	**84.10**
$RRS_{n=100}$	26.80	**94.78**	**94.78**	22.65	**82.90**	**82.90**
$RRS_{n=200}$	38.63	**96.38**	**96.38**	28.61	86.07	86.07
$RRS_{n=500}$	56.31	**97.05**	**97.05**	45.49	91.86	91.86
$RRS_{n=1000}$	66.91	**97.18**	**97.18**	51.52	**92.74**	**92.74**
$RRS_{n=10000}$	73.27	**97.31**	**97.31**	45.21	88.29	88.29
	Prec					
$RRS_{n=10}$	**25.77**	14.84	14.84	**27.20**	14.80	14.80
$RRS_{n=20}$	**24.87**	14.34	14.34	**24.84**	13.77	13.77
$RRS_{n=50}$	**22.84**	13.63	13.63	**23.44**	13.87	13.87
$RRS_{n=100}$	**21.02**	13.39	13.39	**20.93**	13.32	13.32
$RRS_{n=200}$	**19.15**	13.22	13.22	**20.12**	13.42	13.42
$RRS_{n=500}$	**16.86**	13.15	13.15	**19.08**	13.47	13.47
$RRS_{n=1000}$	**15.54**	13.13	13.13	**17.92**	13.50	13.50
$RRS_{n=10000}$	14.56	13.13	13.13	**17.59**	13.43	13.43
	Prec@10					
$RRS_{n=10}$	**25.77**	23.74	23.74	**27.14**	23.81	23.81
$RRS_{n=20}$	**25.98**	**25.61**	**25.61**	**25.89**	**25.18**	**25.18**
$RRS_{n=50}$	**26.06**	**26.04**	**26.04**	**28.07**	**27.70**	**27.77**
$RRS_{n=100}$	**29.30**	**29.50**	**29.28**	**28.12**	**27.99**	**27.63**
$RRS_{n=200}$	**30.60**	**30.94**	**30.58**	**30.09**	**30.29**	**29.86**
$RRS_{n=500}$	**32.68**	**33.02**	**32.88**	**31.85**	**32.45**	**31.65**
$RRS_{n=1000}$	**33.47**	**31.44**	**33.74**	**34.02**	**33.02**	**34.17**
$RRS_{n=10000}$	**33.62**	30.29	**33.38**	**32.00**	29.50	**31.80**
	rPrec					
$RRS_{n=10}$	5.61	**18.44**	**18.44**	5.81	**18.62**	**18.62**
$RRS_{n=20}$	8.84	**20.70**	**20.70**	8.48	**19.37**	**19.37**
$RRS_{n=50}$	13.49	**22.20**	**22.23**	13.06	**22.58**	**22.56**
$RRS_{n=100}$	18.05	**23.67**	**23.79**	14.84	**21.90**	**21.83**
$RRS_{n=200}$	**21.58**	**24.39**	**24.38**	17.57	**23.35**	**23.06**
$RRS_{n=500}$	**24.06**	**25.83**	**25.89**	21.28	**25.13**	**25.25**
$RRS_{n=1000}$	**24.86**	**24.89**	**26.46**	22.96	**26.16**	**26.62**
$RRS_{n=10000}$	**25.06**	**24.44**	**26.81**	19.85	23.91	**24.82**
	AvgPrec					
$RRS_{n=10}$	3.58	**13.51**	**13.51**	3.68	**13.51**	**13.51**
$RRS_{n=20}$	5.30	**17.20**	**17.20**	5.00	**15.66**	**15.66**
$RRS_{n=50}$	7.57	**20.16**	**20.19**	8.35	**20.35**	**20.34**
$RRS_{n=100}$	10.59	**22.44**	**22.41**	9.12	**19.52**	**19.54**
$RRS_{n=200}$	13.84	**24.09**	**23.90**	10.86	**21.06**	20.87
$RRS_{n=500}$	18.02	**24.84**	**25.25**	15.55	**23.98**	**24.02**
$RRS_{n=1000}$	20.37	**24.49**	**25.92**	17.81	**24.49**	**25.30**
$RRS_{n=10000}$	21.77	**23.91**	**26.18**	15.22	21.82	22.92

Table A.35: Impact of PAR on filtered RRS (Google, CAL500)

	RRS	+PAR, $k=50$		QC	+PAR, $k=50$	
		$\alpha=10$	$\alpha=100$		$\alpha=10$	$\alpha=100$
	Rec					
$RRS_{n=10}$	5.58	**61.04**	**61.04**	3.94	49.01	49.01
$RRS_{n=20}$	9.13	**75.70**	**75.70**	6.56	**59.89**	**59.89**
$RRS_{n=50}$	15.68	**90.02**	**90.02**	13.23	76.24	76.24
$RRS_{n=100}$	23.66	**94.04**	**94.04**	17.51	75.47	75.47
$RRS_{n=200}$	33.42	**95.13**	**95.13**	22.64	79.01	79.01
$RRS_{n=500}$	47.36	**95.59**	**95.59**	34.22	85.99	85.99
$RRS_{n=1000}$	56.09	**95.85**	**95.85**	36.95	85.74	85.74
$RRS_{n=10000}$	61.43	**95.87**	**95.87**	30.06	79.64	79.64
	Prec					
$RRS_{n=10}$	**26.74**	14.87	14.87	19.95	13.35	13.35
$RRS_{n=20}$	**24.11**	14.25	14.25	**21.67**	13.12	13.12
$RRS_{n=50}$	**22.11**	13.76	13.76	**20.83**	14.08	14.08
$RRS_{n=100}$	**20.63**	13.47	13.47	**20.67**	12.97	12.97
$RRS_{n=200}$	**18.72**	13.33	13.33	**20.71**	13.10	13.10
$RRS_{n=500}$	**17.20**	13.27	13.27	**19.36**	13.68	13.68
$RRS_{n=1000}$	**16.25**	13.26	13.26	**18.61**	13.11	13.11
$RRS_{n=10000}$	**15.45**	13.25	13.25	**21.44**	13.56	13.56
	Prec@10					
$RRS_{n=10}$	**26.80**	**24.24**	**24.24**	19.95	19.71	19.71
$RRS_{n=20}$	**25.42**	**24.24**	**24.24**	**21.32**	**19.14**	**19.14**
$RRS_{n=50}$	**26.55**	**26.04**	**25.97**	**25.06**	**26.33**	**26.26**
$RRS_{n=100}$	**27.99**	**27.27**	**27.41**	**26.23**	**25.90**	**25.97**
$RRS_{n=200}$	**28.63**	**27.84**	**28.13**	**28.30**	**27.91**	**27.48**
$RRS_{n=500}$	**29.28**	**29.21**	**28.71**	**29.26**	**30.29**	**29.93**
$RRS_{n=1000}$	**29.50**	**28.99**	**29.57**	**29.71**	**28.63**	**29.64**
$RRS_{n=10000}$	**30.43**	**27.77**	**30.00**	**29.98**	**28.13**	**28.42**
	rPrec					
$RRS_{n=10}$	5.33	**18.74**	**18.74**	3.73	**15.72**	**15.72**
$RRS_{n=20}$	7.65	**19.74**	**19.74**	6.15	**16.41**	**16.41**
$RRS_{n=50}$	11.88	**21.01**	**21.02**	10.10	**19.77**	**19.71**
$RRS_{n=100}$	15.81	**22.30**	**22.28**	12.32	**20.32**	**20.33**
$RRS_{n=200}$	18.77	**22.80**	**22.68**	14.35	**20.89**	**20.58**
$RRS_{n=500}$	20.77	**23.51**	**23.54**	17.96	**23.80**	**23.59**
$RRS_{n=1000}$	**21.02**	**23.33**	**23.95**	17.93	**23.00**	**23.21**
$RRS_{n=10000}$	**21.12**	**22.32**	**24.07**	15.51	**21.28**	**21.82**
	AvgPrec					
$RRS_{n=10}$	2.78	**13.33**	**13.33**	2.26	**10.70**	**10.70**
$RRS_{n=20}$	4.16	**16.06**	**16.07**	3.53	12.74	12.74
$RRS_{n=50}$	6.34	**19.12**	**19.05**	5.73	**16.56**	**16.57**
$RRS_{n=100}$	8.52	**20.63**	**20.52**	7.22	**17.26**	**17.27**
$RRS_{n=200}$	10.85	**21.54**	**21.40**	8.93	**18.54**	**18.45**
$RRS_{n=500}$	14.15	**22.36**	**22.39**	11.94	**21.07**	**21.22**
$RRS_{n=1000}$	15.93	**22.46**	**22.92**	12.74	**21.08**	**21.44**
$RRS_{n=10000}$	16.96	**21.50**	**22.97**	10.80	19.35	19.61

Table A.36: Impact of PAR on filtered RRS (exalead, CAL500)

	ψ	+PAR, $k=50$		ANR	+PAR, $k=50$	
		$\alpha=10$	$\alpha=100$		$\alpha=10$	$\alpha=100$
Rec	93.66	**98.50**	**98.50**	93.31	**98.34**	**98.34**
Prec	4.27	3.68	3.68	**4.39**	3.68	3.68
Prec@10	**39.25**	8.55	23.40	**41.05**	9.30	25.40
rPrec	30.78	12.57	26.22	**31.52**	13.42	26.91
AvgPrec	25.97	11.72	21.41	**26.64**	12.35	22.29
AUC	27.09	12.98	22.67	**27.89**	13.70	23.66

Table A.37: Impact of PAR on filtered PseudoDoc (Google, c35k)

	ψ	+PAR, $k=50$		ANR	+PAR, $k=50$	
		$\alpha=10$	$\alpha=100$		$\alpha=10$	$\alpha=100$
Rec	87.20	**96.09**	**96.09**	86.39	95.83	95.83
Prec	5.52	3.73	3.73	**5.68**	3.74	3.74
Prec@10	**34.95**	12.00	**26.65**	**35.75**	13.05	**28.25**
rPrec	26.72	15.26	24.86	**27.28**	16.17	25.51
AvgPrec	21.16	12.77	19.49	**21.76**	13.45	**20.33**
AUC	22.29	13.95	20.62	**22.92**	14.64	**21.48**

Table A.38: Impact of PAR on filtered PseudoDoc (exalead, c35k)

	ψ	+PAR, $k=50$		ANR	+PAR, $k=50$	
		$\alpha=10$	$\alpha=100$		$\alpha=10$	$\alpha=100$
Rec	81.15	**98.83**	**98.83**	80.79	**98.83**	**98.83**
Prec	**14.50**	13.34	13.34	**14.55**	13.34	13.34
Prec@10	**30.72**	27.63	**31.01**	**31.73**	28.06	**32.16**
rPrec	**25.77**	**24.15**	**26.64**	**25.76**	**24.29**	**26.50**
AvgPrec	22.66	**23.89**	**25.89**	22.84	**23.91**	**25.90**
AUC	24.12	**25.36**	**27.41**	24.33	**25.39**	**27.47**

Table A.39: Impact of PAR on filtered PseudoDoc (Google, CAL500)

	ψ	+PAR, $k=50$		ANR	+PAR, $k=50$	
		$\alpha=10$	$\alpha=100$		$\alpha=10$	$\alpha=100$
Rec	70.52	**97.97**	**97.97**	69.42	**97.97**	**97.97**
Prec	**15.08**	13.38	13.38	**15.13**	13.38	13.38
Prec@10	**26.47**	**26.33**	**26.62**	**27.84**	**27.70**	**27.99**
rPrec	**21.31**	**22.47**	**22.92**	**22.39**	**22.94**	**24.06**
AvgPrec	17.57	**22.07**	**22.56**	18.57	**22.72**	**23.65**
AUC	19.21	**23.54**	**24.24**	20.09	**24.09**	**25.21**

Table A.40: Impact of PAR on filtered PseudoDoc (exalead, CAL500)

A.2.4 Impact of Audio-Based Combination on Long-Tail Retrieval

	orig	prun	+PAR	orig	prun	+PAR
	Rec			**Prec**		
$RRS_{n=10}$	2.18	1.02	**5.40**	**30.15**	**27.95**	6.80
$RRS_{n=20}$	3.74	1.70	**9.01**	**29.02**	**28.08**	6.79
$RRS_{n=50}$	7.17	3.41	**16.67**	**27.61**	**27.26**	6.52
$RRS_{n=100}$	12.72	6.35	**25.57**	**25.99**	**25.72**	6.01
$RRS_{n=200}$	18.67	9.19	**34.86**	**23.77**	**23.39**	5.56
$RRS_{n=500}$	29.31	14.15	**49.27**	**20.12**	**19.65**	5.06
$RRS_{n=1000}$	40.38	19.32	**60.58**	**16.88**	**16.46**	4.68
$RRS_{n=10000}$	**80.50**	38.70	**86.38**	**7.29**	**7.21**	3.85
$PseudoDoc$	**93.66**	45.00	90.32	**4.27**	**4.26**	3.64
	AvgPrec			**Overlap**		
$RRS_{n=10}$	1.19	0.56	**1.09**	**100.00**	49.19	51.67
$RRS_{n=20}$	1.84	0.91	**1.81**	**100.00**	48.13	51.97
$RRS_{n=50}$	3.24	1.60	**3.13**	**100.00**	48.64	55.73
$RRS_{n=100}$	**5.54**	2.98	**5.14**	**100.00**	49.45	60.06
$RRS_{n=200}$	**8.23**	4.28	**6.93**	**100.00**	48.71	63.87
$RRS_{n=500}$	**12.39**	6.30	**9.54**	**100.00**	48.24	70.49
$RRS_{n=1000}$	**16.10**	7.97	**11.36**	**100.00**	47.94	76.12
$RRS_{n=10000}$	**29.98**	14.61	16.23	**100.00**	48.60	90.32
$PseudoDoc$	**25.97**	12.80	13.05	**100.00**	48.84	93.33

Table A.41: Impact of audio-combination on long-tail retrieval (Google, c35k)

	orig	prun	+PAR	orig	prun	+PAR
	Rec			**Prec**		
$RRS_{n=10}$	1.73	1.10	**4.65**	**28.42**	**26.90**	6.70
$RRS_{n=20}$	2.80	1.68	**7.44**	**28.05**	**27.22**	6.44
$RRS_{n=50}$	5.46	3.23	**14.44**	**26.43**	**26.50**	6.41
$RRS_{n=100}$	8.71	5.00	**21.77**	**25.06**	**25.57**	6.03
$RRS_{n=200}$	13.40	7.53	**30.63**	**23.01**	**23.31**	5.68
$RRS_{n=500}$	23.19	12.44	**44.47**	**20.41**	**20.52**	5.23
$RRS_{n=1000}$	34.01	18.15	**56.23**	**17.97**	**18.15**	4.86
$RRS_{n=10000}$	72.48	37.95	**82.50**	**8.81**	**8.99**	3.93
$PseudoDoc$	**87.20**	45.37	87.04	**5.52**	**5.72**	3.71
	AvgPrec			**Overlap**		
$RRS_{n=10}$	0.93	0.57	**1.08**	**100.00**	54.50	55.67
$RRS_{n=20}$	1.38	0.87	**1.58**	**100.00**	53.87	56.56
$RRS_{n=50}$	2.42	1.54	**2.80**	**100.00**	53.84	59.69
$RRS_{n=100}$	3.72	2.34	**4.09**	**100.00**	51.93	60.59
$RRS_{n=200}$	5.34	3.36	**5.71**	**100.00**	51.37	64.01
$RRS_{n=500}$	8.42	5.08	**8.15**	**100.00**	50.96	69.78
$RRS_{n=1000}$	**11.96**	7.12	**10.54**	**100.00**	51.02	75.02
$RRS_{n=10000}$	**23.97**	13.42	15.76	**100.00**	50.59	87.81
$PseudoDoc$	**21.16**	11.54	12.81	**100.00**	50.51	91.03

Table A.42: Impact of audio-combination on long-tail retrieval (exalead, c35k)

	orig	prun	+PAR	orig	prun	+PAR
	Rec			**Prec**		
$RRS_{n=10}$	5.96	3.13	**39.47**	**25.77**	**25.07**	15.72
$RRS_{n=20}$	10.19	4.79	**56.59**	**24.87**	**24.26**	14.85
$RRS_{n=50}$	17.99	8.17	**76.57**	**22.84**	**22.32**	14.07
$RRS_{n=100}$	26.80	12.11	**87.07**	**21.02**	**20.25**	13.64
$RRS_{n=200}$	38.63	17.19	**92.44**	**19.15**	**18.49**	13.41
$RRS_{n=500}$	56.31	25.70	**94.32**	**16.86**	**16.40**	13.25
$RRS_{n=1000}$	66.91	31.13	**94.86**	**15.54**	**15.19**	13.20
$RRS_{n=10000}$	73.27	34.39	**94.97**	**14.56**	**14.22**	13.18
$PseudoDoc$	81.15	38.34	**97.18**	**14.50**	**14.40**	13.37
	AvgPrec			**Overlap**		
$RRS_{n=10}$	3.58	2.05	**9.54**	**100.00**	51.15	69.08
$RRS_{n=20}$	5.30	2.98	**12.77**	**100.00**	49.78	76.21
$RRS_{n=50}$	7.57	4.20	**16.17**	**100.00**	48.96	85.90
$RRS_{n=100}$	10.59	5.57	**18.24**	**100.00**	48.94	91.80
$RRS_{n=200}$	13.84	6.86	**19.08**	**100.00**	48.89	94.94
$RRS_{n=500}$	**18.02**	8.79	**19.58**	**100.00**	49.04	96.76
$RRS_{n=1000}$	**20.37**	10.13	**19.73**	**100.00**	49.24	97.19
$RRS_{n=10000}$	**21.77**	11.01	**19.84**	**100.00**	49.43	97.24
$PseudoDoc$	**22.66**	11.46	**19.95**	**100.00**	49.15	98.23

Table A.43: Impact of audio-combination on long-tail retrieval (Google, CAL500)

	orig	prun	+PAR	orig	prun	+PAR
	Rec			**Prec**		
$RRS_{n=10}$	5.58	3.14	**40.25**	**26.74**	**25.87**	15.27
$RRS_{n=20}$	9.13	4.53	**55.84**	**24.11**	**23.43**	14.68
$RRS_{n=50}$	15.68	7.75	**76.77**	**22.11**	**21.47**	14.17
$RRS_{n=100}$	23.66	11.65	**86.59**	**20.63**	**20.20**	13.85
$RRS_{n=200}$	33.42	17.04	**90.39**	**18.72**	**18.44**	13.55
$RRS_{n=500}$	47.36	23.95	**92.02**	**17.20**	**17.04**	13.43
$RRS_{n=1000}$	56.09	28.24	**92.18**	**16.25**	**16.04**	13.40
$RRS_{n=10000}$	61.43	30.80	**92.30**	**15.45**	**15.20**	13.39
$PseudoDoc$	70.52	35.01	**95.86**	**15.08**	**15.63**	13.54
	AvgPrec			**Overlap**		
$RRS_{n=10}$	2.78	1.69	**9.21**	**100.00**	48.40	64.96
$RRS_{n=20}$	4.16	2.52	**12.19**	**100.00**	48.84	74.13
$RRS_{n=50}$	6.34	3.66	**15.61**	**100.00**	49.11	84.99
$RRS_{n=100}$	8.52	4.69	**17.27**	**100.00**	49.61	90.80
$RRS_{n=200}$	10.85	6.02	**18.02**	**100.00**	49.71	93.67
$RRS_{n=500}$	14.15	7.71	**18.53**	**100.00**	49.36	94.85
$RRS_{n=1000}$	**15.93**	8.62	**18.58**	**100.00**	49.43	95.08
$RRS_{n=10000}$	**16.96**	9.05	**18.41**	**100.00**	49.39	95.10
$PseudoDoc$	**17.57**	9.57	**18.81**	**100.00**	49.67	97.33

Table A.44: Impact of audio-combination on long-tail retrieval (exalead, CAL500)

Curriculum Vitae of the Author

Dipl.-Ing. Peter Knees

Year of Birth: 1982
Nationality: Austrian
Languages: German, English, Italian

Current Affiliation:

Johannes Kepler University Linz
Department of Computational Perception
Altenberger Str. 69
4040 Linz
Austria

Phone: +43 732 2468 1513
Fax: +43 732 2468 1520

Email: peter.knees@jku.at
Homepage: http://www.cp.jku.at/people/knees

Education

2005 – 2010 Ph.D. Study in Computer Science at Johannes Kepler University Linz

February 2005 Graduation in Computer Science (Informatik) at Vienna University of Technology, Academic Degree: Diplom-Ingenieur (M.Sc. equivalent)

since 2004 Diploma Study in Psychology at University of Vienna

2000 - 2005 Diploma Study in Computer Science (Informatik) at Vienna University of Technology

June 2000 Graduation from High School (Matura mit ausgezeichnetem Erfolg); "Pass with High Distinction", Final Result: 1.0 (Maximum: 1.0)

1992 - 2000 High School: Bundesrealgymnasium Rahlgasse, Vienna

Practical Experience

2009 Civilian National Service (Zivildienst) at Verein Wiener Sozialprojekte – ChEck iT!

2008 – 2010 Project Assistant at the Department of Computational Perception at Johannes Kepler University Linz. Involved in the project *Music Retrieval Beyond Simple Audio Similarity* funded by the Austrian National Science Fund (FWF)

2005 – 2008 Project Assistant at the Department of Computational Perception at Johannes Kepler University Linz. Involved in the project *Operational Models of Music Similarity for Music Information Retrieval* funded by the Austrian National Science Fund (FWF)

since 2005 Conceptual Design, Visual Design, and Maintenance of the Webpage of the Department of Computational Perception at JKU Linz

2004 Tutor for "Übersetzerbau" (Compiler Construction) at Vienna University of Technology

2001 - 2005 Tutor for "Einführung in das Programmieren" (Introduction to Programming) at Vienna University of Technology

July 2001 Internship at Siemens Software-Engineering Group Vienna

1998 - 2000 Administrator of Public Computer Network for Students at BRG Rahlgasse

Awards and Prizes

2007 Best Student Paper Award at the 6th International Conference on Mobile and Ubiquitous Multimedia (MUM 2007) for the Paper *One-Touch Access to Music on Mobile Devices* by D. Schnitzer, T. Pohle, P. Knees, and G. Widmer

2006 Runner-up for Best Paper Award at the ACM Multimedia 2006 for the Paper *An Innovative Three-Dimensional User Interface for Exploring Music Collections Enriched with Meta-Information from the Web* by P. Knees, M. Schedl, T. Pohle, and G. Widmer

2003 and 2004 Received **Achievement Scholarships** (Leistungsstipendien) from Vienna University of Technology

Invited Talks

December 2009 Invited Talk (jointly with Markus Schedl) on *Context-Based Music Similarity Estimation*, 3rd Workshop on Learning the Semantics of Audio Signals, Graz, Austria.

April 2008 Seminar Talk on *Integrating Context and Content to Search for Music via Natural Language Queries*, Queen Mary University, London, UK.

Scientific Services

Program Chair and Local Organizer of the *8th International Workshop on Adaptive Multimedia Retrieval* (AMR 2010), Linz, Austria.

Program Chair and Co-Organizer of the *3rd International Workshop on Advances in Music Information Research* (AdMIRe 2011), Barcelona, Spain.

Program Chair and Co-Organizer of the *2nd International Workshop on Advances in Music Information Research* (AdMIRe 2010), Singapore.

Program Chair and Co-Organizer of the *International Workshop on Advances in Music Information Research* (AdMIRe 2009), San Diego, USA.

Program Committee Member of the *Workshop on Music Recommendation and Discovery* (Womrad 2010), Barcelona, Spain.

Technical Program Committee Member of *ACM Multimedia Interactive Arts Program 2010*, Florence, Italy.

Technical Program Committee Member of *ACM Multimedia Interactive Arts Program 2009*, Beijing, China.

Reviewer for international journals such as *Machine Learning*, *IEEE MultiMedia*, *IEEE Transactions on Multimedia*, *IEEE Journal of Selected Topics in Signal Processing*, *IEEE Transactions on Knowledge and Data Engineering*, *International Journal on Digital Libraries*, and *Journal of New Music Research*, as well as for international conferences such as *International Joint Conference on Artificial Intelligence*, *International Conference on Music Information Retrieval*, *International Conference on Semantics and Digital Media Technology*, and *IEEE Visualization* conference.

Publications

Journal Publications

A Music Information System Automatically Generated via Web Content Mining Techniques. M. Schedl, G. Widmer, P. Knees, and T. Pohle. Information Processing & Management (to appear).

Exploring Music Collections in Virtual Landscapes. P. Knees, M. Schedl, T. Pohle, and G. Widmer. IEEE MultiMedia, vol. 14, no. 3, pp. 46–54, 2007.

"Reinventing The Wheel": A Novel Approach to Music Player Interfaces. T. Pohle, P. Knees, M. Schedl, E. Pampalk, and G. Widmer. IEEE Transactions on Multimedia, vol. 9, no. 3, pp. 567–575, 2007.

Automatic Classification of Musical Artists based on Web-Data. P. Knees, E. Pampalk, and G. Widmer. ÖGAI Journal, vol. 24, no. 1, pp. 16–25, 2005.

Peer-Reviewed Conference Publications (Selection)

Supervised and Unsupervised Web Document Filtering Techniques to Improve Text-Based Music Retrieval. P. Knees, M. Schedl, T. Pohle, K. Seyerlehner, and G. Widmer. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, Utrecht, Netherlands, 2010.

Context-based Music Similarity Estimation. M. Schedl and P. Knees. In *Proceedings of the 3rd International Workshop on Learning the Semantics of Audio Signals*, Graz, Austria, 2009.

Augmenting Text-Based Music Retrieval with Audio Similarity. P. Knees, T. Pohle, M. Schedl, D. Schnitzer, K. Seyerlehner, and G. Widmer. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, Kobe, Japan, 2009.

sound/tracks: Real-Time Synaesthetic Sonification and Visualisation of Passing Landscapes. T. Pohle, P. Knees, and G. Widmer. In *Proceedings of the 16th ACM International Conference on Multimedia.* Vancouver, Canada, 2008.

A Document-centered Approach to a Natural Language Music Search Engine. P. Knees, T. Pohle, M. Schedl, D. Schnitzer, and K. Seyerlehner. In *Proceedings of the 30th European Conference on Information Retrieval*, Glasgow, UK, 2008.

One-Touch Access to Music on Mobile Devices. D. Schnitzer, T. Pohle, P. Knees, and G. Widmer. In *Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia*, Oulu, Finland, 2007. (**Best Student Paper Award**)

Browsing the Web Using Stacked Three-Dimensional Sunbursts to Visualize Term Co-Occurrences and Multimedia Content. M. Schedl, P. Knees, G. Widmer, K. Seyerlehner, and T. Pohle. In *Proceedings of the IEEE Visualization 2007*, Sacramento, USA, 2007.

The Quest for Ground Truth in Musical Artist Tagging in the Social Web Era. G. Geleijnse, M. Schedl, and P. Knees. In *Proceedings of the 8th International Conference on Music Information Retrieval*, Vienna, Austria, 2007.

Search & Select - Intuitively Retrieving Music from Large Collections. P. Knees. In *Proceedings of the 8th International Conference on Music Information Retrieval*, Vienna, Austria, 2007.

A Music Search Engine Built upon Audio-based and Web-based Similarity Measures. P. Knees, T. Pohle, M. Schedl, and G. Widmer. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Amsterdam, Netherlands, 2007.

Searching for Music using Natural Language Queries and Relevance Feedback. P. Knees and G. Widmer. In *Proceedings of the 5th International Workshop on Adaptive Multimedia Retrieval*, Paris, France, 2007.

Automatically Describing Music on a Map. P. Knees, T. Pohle, M. Schedl, and G. Widmer. In *Proceedings of the 1st Workshop on Learning the Semantics of Audio Signals*, Athens, Greece, 2006.

Combining Audio-based Similarity with Web-based Data to Accelerate Automatic Music Playlist Generation. P. Knees, T. Pohle, M. Schedl, and G. Widmer. In *Proceedings of the 8th ACM SIGMM International Workshop on Multimedia Information Retrieval*, Santa Barbara, USA, 2006.

An Innovative Three-Dimensional User Interface for Exploring Music Collections Enriched with Meta-Information from the Web. P. Knees, M. Schedl, T. Pohle, and G. Widmer. In *Proceedings of the 14th ACM International Conference on Multimedia*, Santa Barbara, USA, 2006. (**Runner-Up for Best Paper Award**)

Improving Prototypical Artist Detection by Penalizing Exorbitant Popularity. M. Schedl, P. Knees, and G. Widmer. In *Proceedings of the 3rd International Symposium on Computer Music Modeling and Retrieval*, Pisa, Italy, 2005.

Multiple Lyrics Alignment: Automatic Retrieval of Song Lyrics. P. Knees, M. Schedl, and G. Widmer. In *Proceedings of the 6th International Conference on Music Information Retrieval*, London, UK, 2005.

Artist Classification with Web-based Data. P. Knees, E. Pampalk, and G. Widmer. In *Proceedings of the 5th International Conference on Music Information Retrieval*, Barcelona, Spain, 2004.