

APPLICATION OF TEXT-BASED METHODS OF ANALYSIS
TO SYMBOLIC MUSIC

by

Jacek Wołkowicz

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
March 2013

© Copyright by Jacek Wołkowicz, 2013

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled “APPLICATION OF TEXT-BASED METHODS OF ANALYSIS TO SYMBOLIC MUSIC” by Jacek Wołkowicz in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dated: March 20, 2013

External Examiner:

Ichiro Fujinaga

Research Supervisor:

Vlado Kešelj

Examining Committee:

Jason Brown

Nauzer Kalyaniwalla

Departmental Representative:

Michael McAllister

DALHOUSIE UNIVERSITY

DATE: March 20, 2013

AUTHOR: Jacek Wolkowicz

TITLE: APPLICATION OF TEXT-BASED METHODS OF ANALYSIS TO
SYMBOLIC MUSIC

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: Ph.D.

CONVOCATION: May

YEAR: 2013

Permission is herewith granted to Dalhousie University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions. I understand that my thesis will be electronically available to the public.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

The author attests that permission has been obtained for the use of any copyrighted material appearing in the thesis (other than brief excerpts requiring only proper acknowledgement in scholarly writing), and that all such use is clearly acknowledged.

Signature of Author

Table of Contents

List of Tables	viii
List of Figures	ix
Abstract	xii
Chapter 1 Introduction	1
1.1 Challenges for Research in Computational Musicology	3
1.2 Levels of Computational Music Research	9
1.3 Importance of Symbolic Music Research	11
1.4 Research Questions	13
1.5 Contributions	15
1.6 Publications	17
1.7 Overview of the Dissertation	19
Chapter 2 Background	21
2.1 Recorded Audio	21
2.2 Symbolic Representations	23
2.2.1 MIDI Protocol	26
2.3 Format Conversions	28
2.4 Monophonic Music Encoding	32
2.4.1 Pitch Encodings	34
2.4.2 Rhythm Encodings	41
2.5 Analysis of Symbolic Music	45
2.5.1 String Matching	46
2.5.2 Geometric Approaches	50
2.5.3 N-gram Approaches	51
2.5.4 Hierarchical Approaches	54
2.5.5 Other Approaches	55

2.6	Linguistic Parallels in Music Research	56
2.6.1	History of NLP and Music Research	58
2.6.2	Music Linguistic Theories	60
2.6.3	Future of Music Research	61
Chapter 3	Quantitative Analysis of Symbolic Music	63
3.1	N-gram-based Approach to Symbolic Music	64
3.2	Available Corpora	67
3.2.1	Symbolic Music Corpora	68
3.2.2	Textual Corpora	73
3.3	PERL MIDI::Corpus Module	74
3.3.1	Synopsis	75
3.3.2	Corpus Loading	75
3.3.3	Corpus Status	78
3.3.4	Playback	78
3.3.5	Data Functions	79
3.3.6	Analysis Functions	80
3.3.7	Similarity Measures	83
3.4	Statistical Analysis of Music Corpora	84
3.4.1	Size Statistics	84
3.4.2	Distribution of Intervals and IORs	84
3.4.3	Zipf's Law for n-grams	93
3.4.4	Entropy Analysis	95
3.4.5	Complexity and Relations Between Datasets	98
3.5	Composer Classification of Symbolic Corpora	103
3.5.1	Methodology	104
3.5.2	Results	108
3.5.3	Summary of Composer Classification Experiment	116
3.6	Symbolic Music Similarity	117
3.6.1	Background	118
3.6.2	Methodology	121
3.6.3	Analysis Based on RISM Data	126
3.6.4	Conclusions	133
3.7	Summary of the Chapter	134

Chapter 4	Using Genetic Algorithms for Evolving Excerpts with Corpus-Based Fitness Evaluation	136
4.1	Background	136
4.1.1	Previous Work	137
4.2	Methodology	139
4.2.1	GA Design Decisions	141
4.3	Experiments	143
4.3.1	N-gram Statistics	143
4.3.2	Generational GA	146
4.3.3	Steady-State GA	147
4.3.4	Diversity by Multi-Objective Optimization	149
4.3.5	Results	152
4.4	Issues and Possible Extensions	152
Chapter 5	Symbolic Music Visualization	155
5.1	Existing Approaches to Music Visualization	156
5.2	Methodology for Symbolic Music Visualization	160
5.2.1	Use of Similarity Matrices as a Layout for Visualization	161
5.2.2	Dealing with Multiple Layers and Tracks	162
5.2.3	Comparison Function for Music Excerpts	163
5.2.4	Dealing with Layer Overload	164
5.2.5	Algorithm for Matrix Generation	167
5.3	Visualization Package	167
5.4	Examples and Features of Proposed Visualization Scheme	172
5.5	Future Work	177
Chapter 6	Structural Analysis of Symbolic Music	178
6.1	Motivation	178
6.2	Related Work	180
6.3	Methodology	183
6.3.1	Similarity Measure	184
6.3.2	Similarity Matrices Adjustments for Structural Analysis	193

6.3.3	Novelty Detection Function	197
6.3.4	Modified Dijkstra Algorithm	198
6.3.5	Automatic Structure Inference	199
6.4	Analysis of the Results	201
6.5	Conclusions	211
Chapter 7	Conclusions	212
7.1	Presented Work	212
7.2	Impact and Implications	215
7.3	Future Work	216
Bibliography	218

List of Tables

2.1	Score formats.	25
2.2	NLP levels	57
3.1	Size statistics of analyzed corpora.	85
3.2	Entropy values of symbols in various corpora.	97
3.3	ADR results for SMS 2011 task.	132
3.4	FP10 results for SMS 2011 task.	132
3.5	Per query FP10 results for SMS 2011 task.	133
4.1	The samples of resulting melodies.	153
6.1	Significance of the obtained results for coarse segmentation. . .	209
6.2	Significance of the obtained results for fine segmentation. . . .	210

List of Figures

1.1	Levels of music information retrieval.	10
2.1	Human hearing range.	22
2.2	Spectral images of score samples.	30
2.3	Pitch encodings.	38
2.4	Computation of Levenshtein edit distance	48
2.5	Group edit operations shown on a grid graph	50
3.1	N-gram extraction process.	66
3.2	Distribution of melodic intervals in all corpora.	87
3.3	Distribution of rhythmic IORs in all corpora.	88
3.4	Distribution of intervals and IORs in <code>essen</code> , <code>pcd_ex</code> , <code>string</code> , and <code>wtk</code> datasets.	90
3.5	Distribution of intervals and IORs in <code>essen</code> and <code>rism</code>	91
3.6	Distribution of intervals and IORs in <code>beatles</code> , <code>wtk</code> , and <code>pcd</code>	92
3.7	Zipf's law for music and text features	96
3.8	Factors influencing entropy estimation.	99
3.9	Compressibility of symbolic music datasets.	101
3.10	Similarities between corpora based on Kolmogorov complexity.	102
3.11	Accuracy of classification for <code>pcd</code> dataset.	108
3.12	Accuracy of classification for <code>string</code> and <code>string_rh</code> datasets.	110
3.13	Accuracy of 'Mozart' and 'Haydn' classes for <code>string</code> dataset.	111
3.14	Accuracy of 'Mozart' and 'Haydn' classes for <code>string_rh</code> dataset.	112
3.15	The comparison of similarity measures with cosine similarity method using various feature extraction method.	128

3.16	The influence of quantization approaches.	128
3.17	Comparison of three similarity measures for fine melodic interval features.	129
3.18	Comparison of weighting schemes for fine melodic interval features and cosine similarity.	130
4.1	Two children created using different representations for standard crossover.	141
4.2	Distribution of n-gram counts, frequencies and tf.idf scores. . .	145
4.3	Generational GA evolving 100 melodies over 500 generations. .	146
4.4	Steady-state GA evolving run over 100 epochs.	148
4.5	Steady-state GA evolving run over 100 epochs averaged over 100 runs with different mutation probabilities.	150
4.6	Converging towards Pareto-front at epochs 1, 10, 100 and 1000.	151
5.1	Bach prelude C major - an excerpt visualized using three methods.	158
5.2	Visualized excerpt from Bach prelude C sharp Major	161
5.3	Plethora of layers in 5 voci C sharp minor Bach's Fugue. . . .	165
5.4	Excerpt from C sharp minor Bach's Fugue displayed 3rd voce comparison only.	165
5.5	Excerpt from C sharp minor Bach's Fugue filtered by the theme.	166
5.6	Excerpt from C sharp minor Bach's Fugue filtered by the counterpoint.	166
5.7	Main window of the visualization program.	170
5.8	J.S. Bach prelude in C sharp major — structure and textures.	173
5.9	Fugue in C major filtered by the main theme pattern.	174
5.10	Three occurrences of the same theme, played with three different tempi.	175
5.11	Fugue in E flat major filtered by the main theme pattern with augmented themes.	175

5.12	Visualization of the Beatles “The Magical Mystery Tour”. . .	176
6.1	The method of obtaining similarity matrices from symbolically represented music pieces.	185
6.2	Indications of a good structure analysis for similarity measures.	186
6.3	Similarity matrices for test input using Hamming and Leveshtein edit distances.	191
6.4	Similarity matrices for test input using bag-of-words approaches.	192
6.5	Similarity matrices for Bach Prelude C# major with different similarity measures.	194
6.6	Similarity matrices for the Beatles’ “The Magical Mystery Tour” with different similarity measures.	195
6.7	An example showing how layers are joined to form one vector of features.	197
6.8	Structure inference from self-similarity matrices step by step. .	200
6.9	Illustration how group ID’s propagate and are inferred from the shortest path.	201
6.10	Structural segmentation of eleven pieces from IA corpus. . . .	205
6.11	Comparison of Precision, Recall and F-measure results.	207
6.12	Precision — Recall graphs indicating performance of compared algorithms.	208

Abstract

This dissertation features methods of analyzing symbolic music, focused on n-gram-based approaches, as this representation resembles the most text and natural languages. The analysis of similarities between several text and music corpora is accompanied with implementation of text-based methods for problems of composer classification and symbolic music similarity definition. Both problems contain thorough evaluation of performance of the systems with comparisons to other approaches on existing testbeds. It is also described how one can use this symbolic representation in conjunction with genetic algorithms to tackle problems like melody generation. The proposed method is fully automated, and the process utilizes n-gram statistics from a sample corpus to achieve it. A method of visualization of complex symbolic music pieces is also presented. It consist of creating a self similarity matrix of a piece in question, revealing dependencies between voices, themes and sections, as well as music structure. A fully automatic technique of inferring music structure from these similarity matrices is also presented The proposed structure analysis system is compared against similar approaches that operate on audio data. The evaluation shows that the presented structure analysis system outperformed significantly all audio-based algorithms available for comparison in both precision and recall.

Chapter 1

Introduction

Beginning with the computer revolution, new opportunities have emerged for music artists and researchers. When some started to execute computational problems on large but slow, by today's standards, mainframe computers others started using them to generate early computer music. This was the starting point for thinking how computers might be used to process and analyze music [102]. Music, similarly to human speech, accompanied human evolution from its beginning, therefore deep understanding of music can allow us to understand better human cognition. Since computers are now the best tools for any kind of complicated and strenuous analysis, musicians and musicologists joined computer science society and created such areas of research like computer music, music information retrieval or music analysis and modelling.

On the other hand, people store large amounts of music on their computers, and more music is available to everybody through the Internet services. These facts make the problem of processing musical data even more important. Musical data is still mostly treated as unstructured binary data left on the same shelf with images, movies, or executables, unlike textual data, which is easy to process, search, or index, and driven by a large number of available computer-aided techniques provided by natural language processing, information retrieval or text data mining like classification, analysis, generation, summarization, indexing, searching and many more.

Additionally, the computational theory of music is an important area of research because of music's mathematical foundations, as well as the fact that computer science, or the theory of computation in general, descended from mathematics and is highly related with it. In ancient Greece, Pythagoras thought that we are all surrounded with noises and tones of various frequencies and intensities, but our mind is not noticing them, because we have just got used to them. Cosmos was also supposed

to make such sounds, but our brains were not able to perceive them. According to Pythagoras, music was one of the four foundations of wisdom, along with arithmetic, astronomy and geometry and each of his apprentices was required to master all four of them. Pythagoreans thought that the world was created from chaos with sound and harmony and according to music rules [160]. Music harmony, which summarizes music from the theoretical perspective, bases its foundations in acoustics, which explores physical and mathematical features of sound.

The aim of this research is to investigate if music can be treated as a natural language and processed in the similar way as text is being processed. This parallel have been used in the previous work [216], where a solution to composer classification problem have been proposed, and here, it is further extended to the problem of unveiling music structure. Showing that music and text are very closely related will be beneficial for both domains of music research and natural language processing, as it would be much easier to port existing algorithms and approaches between them. Although there are substantial differences between written text and music, they have many features in common. The progress in music visualization and automatic navigation makes music more accessible and browsable as text is. Some already existing music visualization systems help users better perceive and navigate through music both on a corpus scale (through various music searching methods in large corpora using music information retrieval techniques), and locally (by aiding user experience with particular music piece using various music analysis, processing and visualization tools).

We developed techniques that aid navigation in music and tested them against existing methods operating on low-level audio representation. Our hypothesis is that although music represented in a symbolic form might be less appealing to the end users, it gives much cleaner information for algorithms, hence allowing for more accurate and thorough analysis. If this holds true, embedding symbolically represented music in audio files would become a necessity for music publishers in the future if they want to benefit from applications, such as ones presented in this dissertation.

1.1 Challenges for Research in Computational Musicology

Research in the field of computational musicology is strongly trans-disciplinary. Application of computer science methods to music requires incorporating some musicological or signal processing background knowledge about the data into the process. Similarly, from the musicology point of view, solving computational problems means that one has to know computer science techniques and algorithms or programming in general. This makes problems of processing music media unique in a sense, that they usually require a targeted approach or special algorithms developed just for those problems. However, one can also point out certain similarities with other areas which often escape the need of coming up with custom approaches and allows using known algorithms for new problems. I will now list major areas that require computational approach in music domain. However, since music research is such a multifaceted field, a different perspective would give a slightly different list.

Music Representation. This area deals with a fundamental question, how to digitally represent music data, and it links many research domains that define trans-disciplinarity of music research including Computer Science, Musicology, Library Science, Cognitive Science [56] and Audio Engineering (compression). The main issues are how to choose proper information, how to encode it so that it allows for more accurate processing, or provide new information to create opportunities for solving new problems. Since this usually leads to defining new standards [7, 8], and one should not change them frequently, it has to be done with great care.

Search. To create an effective and efficient music information retrieval system was the initial goal of the researchers coming from computer science domain into computational music, hence the commonly used name for the entire domain, Music Information Retrieval. Successful models have been proposed. They use symbolically represented corpora, with querying and indexing approaches based on n-grams [33, 41, 107, 193]. It is closely related to textual information retrieval as it uses similar techniques to

achieve similar goals. Orio and Neve [142] pointed out that simple n-gram-based approach yields much better results than more sophisticated models based on a priori music knowledge.

Not all problems in the music search domain have text processing origin. Lots of attention has been paid to Query By Humming (QBH) systems which incorporate melody extraction techniques from hummed queries to match against the database [56]. There are also audio-based systems that, in most cases, try to solve another problem — Query By Example (QBE), where one searches, using a short audio fragment, for the exact song in a database of audio pieces, that the querying fragment comes from. Lee and Downie [105] concluded from their user study, investigating users information needs, that this is a kind of search that users are most likely willing to do. The most successful example of such system is UK-based Shazam [205,206], with more than 2.5 million songs in their database.

Typke et al. [189] lists 17 different music information retrieval systems where 7 operate on audio data only, 6 on symbolic data only and 3 on both.

In this dissertation, we are elaborating on the n-gram approach to symbolic music, showing how it can be applied to other tasks and fields within computational music research.

Browsing and Navigating. Direct searching for a specific phrase or document is not the only way of seeking information. Issuing and understanding music queries is not that straightforward as it is in textual information retrieval. This forms a place for other approaches that do not require forming the actual query [141]. They can either operate on corpus (entire collection) or opus (individual piece) level.

The first task is called music browsing. An effective music browser should allow user to quickly find relevant pieces by organizing, grouping and linking music content in a way, that is meaningful for the user [65]. Lee and Downie [105] identified, that relations between music pieces are among the most desired information that users are seeking for. A good music browser should adjust to user needs, mitigating the lack of the actual query [141].

On the opus level, it is usually called music navigation, and it helps users to locate interesting parts in music pieces. Possible approaches involve creating music summaries, creating linked description of a given piece, indicating verses, choruses enabling users to navigate consciously within a piece. An example of such system has been developed by Goto [66].

Futrelle and Downie [56] referred to this area as User Interface Design, but in our opinion the most challenging part is the actual retrieval of the dependencies and navigational information from music corpora, not the interface part.

Recommendation. This is currently a very active area, not only in music domain. Orio [141] refers to it as music filtering, because such system can be seen as a filter that adjusts itself to user personal needs, making available to the user only those items, that suit them according to their personal profile. It is often called personalized search, when it also involves the actual query from the user. Recommendations are usually based on either social data associated with the music context (collaborative filtering) or directly on music data and non-social metadata, like lyrics (content-based recommendation). According to the survey by Lee and Downie [105], we are most likely to ask family and friends for help when we search for music or music information, twice as eagerly as anybody else, including music professionals. They also found out that we search the most for music heard from our friends and family places and that we are positive to receive recommendations from them as well. This is why recommendation systems are important, as they address typically social users needs through an automatic way. According to Stenzel and Kamps [174], pure collaborative-based models outperform content-based, which may indicate that lots can be done to improve the music information technology side of the music recommendation task.

Classification. This typical data mining task is present in all domains, and for all sorts of data. In music, classification challenges cover detection of various music aspects including genre, composer, performer, sound type (e.g., speech, music, environmental sound [141]), or instruments used in a recording. According to user surveys,

we would respond positively if the system offered us the ability to listen to similar music, similar artists of similar genre of what we are currently listening to [105]. What makes music classification different and the research unique, is the problem of selecting features used for classifier input data. Typically various audio-based features (spectral and temporal), symbolic features and meta-data are aggregated for better results [56]. What is interesting, while music classification solutions are frequently quite complex systems, studies have showed, that “even untrained listeners are quite good at classifying audio data with very short excerpts (less than 1 sec)” [141], which shows the ability of human perception in this regard versus current state-of-the-art computer systems.

Visualization. Similarly to music browsing and navigating, music visualization can be aimed at opus or corpus level. Its main goal is not to automatically create relations, either between music pieces or within a piece, but rather visualize them and leave them to the users to interpret by themselves. This makes the result highly subjective and therefore not easy to evaluate. The only way to do it directly is through user studies, but they are quite rarely seen in music domain.

Orio [141] motivates the need of corpus scoped visualizations with difficulties that users have organizing their personal collections of music and that a meaningful spatial organization of pieces will allow them to draw correct conclusions about where to locate specific pieces. Those solutions are based usually on calculating similarity between pieces and further reducing dimensionality to fit into several visual dimensions or by visualizing them using self-organizing maps (SOMs). They usually aim to suit non-expert users [141].

On the other hand, the main idea behind opus-based visualization techniques is to create a snapshot representing the content of a music piece, so that one can draw conclusions about the music without hearing it. Isaacson [85] refers to Common Music Notation as a first approach to this task. What those approaches have in common is that they usually replace temporal dimension of music with one spatial dimension,

allowing users to capture the main idea behind a piece in a glimpse, much like we perceive images. Chapter 5 of this dissertation provides our solution to this task.

Audio Transcription. Many areas in music research that have been introduced above, prefer some music representations more than others. Symbolic representation carries cleaner and more accurate information for many tasks, while audio is simply more popular, so for some applications, like recommendations, audio data is assumed at the input. For some tasks, like search, we know quite simple solutions that work well with symbolic data but it is currently not possible to transfer them to audio domain. Audio transcription tries to bridge those two worlds by creating tools to automatically recognize notes from the recording. Although the technology is improving, the accuracy of current state-of-the-art solutions, especially for polyphonic music, is still too small if one wants to use their output for further analysis or even just for search [43, 164]. However, there are at least two ways to mitigate the problem. First one uses score-to-audio or midi-to-audio alignment approach, where both symbolic description and audio data are available a priori. We just need to map temporarily symbolic features onto audio data, which can be done fairly accurately [31, 54, 81, 97]. The second approach circumvents the need of the full transcription using a mid-level representation that gives more higher level information than audio, but does not introduce as many errors as a fully transcribed piece. It is useful in some tasks, like structure analysis [125].

Music Information Discovery. This area embraces the needs and hopes of musicologists that they laid upon the arrival of computer techniques that would allow them to analyze music more quickly and thoroughly. The domain is also called Musical Analysis [56]. According to Honing [78], there has been recently an important shift within musicology from perceiving music mostly as an art and musicology as a tool for analyzing art, into a more scientific field, that deals with music as an acoustic, psychological and cognitive phenomenon. This shift into empirical, computational, and cognitive musicology can be seen as a “cognitive revolution” in the humanities [78].

As a result, the role of formalization have increased, resulting in the emergence of formal, computational theories of music with the first one being Lerdahl and Jackendoff's "Generative theory of Tonal Music" [111]. Many have followed with similar ideas (Narmour [135], Temperley [178]) and with the actual implementations of the original theories [26, 70]. Developing those techniques seems important, even when one does not see any direct applications and benefits, because they actually try to achieve what Natural Language Processing is doing for text. Parsing of written text is an essential part of many high-level applications that require this higher-level understanding of a text they deal with. It seems very likely that we will need music information discovery tools for more sophisticated music applications in the near future, apart from the current applications in musicology.

Music OCR. This domain is a part of image-processing domain, but involves theoretical music knowledge to improve and interpret the results. It is a very important, as many of the existing music scores were published before the computer era, and digitizing them is as vital as digitizing books for libraries. Bainbridge and Bell [4] provide an overview of a typical approach to the problem. According to Rebelo et al. [159], most of the difficulties lie not on the image processing side, but in the interpretation of the results which are still far from perfect, especially for hand-written scores.

Music social networks. Mining music social networks is an important and quickly growing area that combines music data and metadata with user social information. It has been shown by Lee [105], that our musical taste is highly influenced by our peers. They call this social information, relational metadata. There are successful systems, like last.fm, that rely solely on social dependencies and previous listening history in playlist generation. Youtube, which is mostly used as a free music store, uses users browsing patterns to recommend similar songs. What is different, and puts this field in a better position comparing to other areas of music research, is that there are big

datasets of socially generated user data for evaluation and development of socially motivated tools, one of the recent examples being million song dataset challenge [127].

1.2 Levels of Computational Music Research

Computational Music Research (CMR) or Music Information Research (MIR) spans all sorts of levels, on which one can perceive music and analyze music data. It is often called Music Information Retrieval for historical reasons, although one could prefer not to use this term, as it reminds too much of textual Information Retrieval, which in the end limits the perceived scope of the domain to search, and search related problems. In fact, MIR deals with a range of problems, from low-level analysis of audio signals using signal processing techniques, to high-level semantic analysis and interpretation of music pieces. Figure 1.2, taken from Fingerhut [50], depicts the dependencies between various levels, that MIR operates on. They differentiate between internal data sources (blue) and external actors (white), which, through their interactions with the system, become also data providers bringing meta-data (on various levels), usage data or social data. Through application of various techniques, a knowledge is generated (in yellow) gradually pushing acoustic data onto semantic and cognitive level. Similarly, hierarchical approach was presented by Vincent et al. [204], where a dynamic Bayesian network model was proposed spanning all facets of music, with similar dependencies as presented by Fingerhut [50]. Downie [39] in his article about Music Information Retrieval defines a similar distinction between different facets of music information indicating main challenges that come with it:

- **multi-representation:** Each piece of music is represented on different levels in different forms, i.e., audio, sheet music or symbolic representations.
- **multi-culturalism:** Not only music, but also approach to music and aesthetics differ among musical cultures.
- **multi-experience:** Since music is art, and by nature it is very subjective, many basic notions are not as clearly defined as for other domains, like text. For

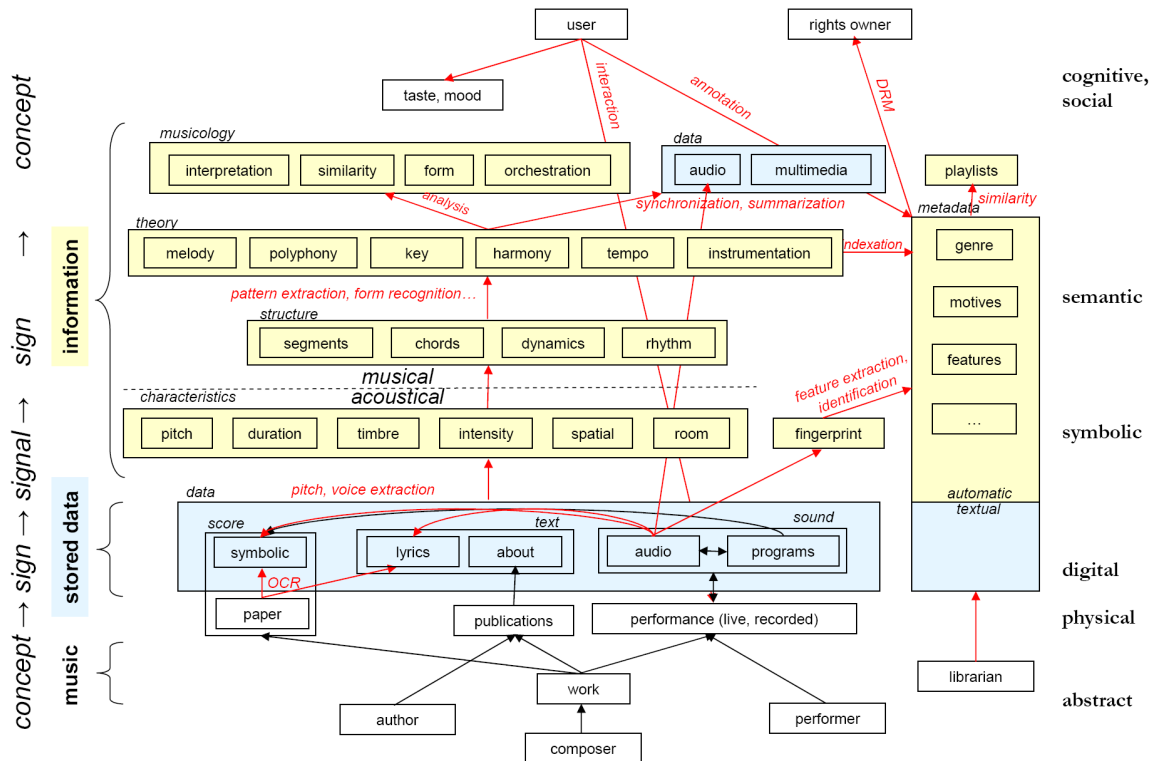


Figure 1.1: Levels of music information retrieval. External data sources are indicated in white, internal in blue, information in yellow and interactions / operations in red arrows. Adapted from [50]

instance music similarity is a very subjective term and everybody can perceive it on different levels. As a result, we can have many different similarity perceptions and all of them are valid.

- multi-disciplinarity:** The entire domain is driven by a compilation of needs of several different research communities (Computer Science, Audio Engineering, Musicology, Library Science, Cognitive Science and Law [56]). Those needs not always match and in some cases, they are even contradictory (e.g., Digital Rights Management (DRM) and data openness), which makes it sometimes impossible to satisfy everybody.

1.3 Importance of Symbolic Music Research

It is relatively easy to indicate, why music research is important in general: there is plenty of music content on the Internet and in personal collections and one needs methods to process, analyze, organize, and search through them, and make the content more semantically meaningful for a user and other applications as well. However, one realizes, that most of the content, that these problems deal with, is in audio form, and most of the solutions to these problems operate in audio domain as well. This raises an important question: why would we ever care about symbolic music. There are actually several good reasons, why it is important to develop solutions for symbolic data.

The lack of symbolic data and other higher-level meta information aligned with audio and available for processing is only temporary, since the general direction for all sorts of digital content is towards combining raw data with its high-level meta-data. For example, currently most images taken with any camera are supplied with EXIF data — a detailed description about the conditions, parameters, even GPS location and direction, where the picture was taken, and even names of people on the actual photo if face detection and face recognition functions are provided with a given camera. The same phenomenon can be observed for music files. If purchased in a good store, they are fully supplied with general information about the song, like artist, title, album, etc. That information is embedded within an MP3 file in a form of an ID3 tag. When information about melodic, harmonic or structural layer would appear as a standard along with audio, then there will be not only plenty of symbolic music corpora, but also corpora of aligned symbolic and audio data for tasks that require both at the same time.

Second reason results from music's dichotomy, or its rather multi-faceted nature (Downie's multi-representation [39]) which states, that every music opus contains information at all those levels of representation, and each level brings some new and important aspects of the given piece. According to Fingerhut [50], music is not only recorded music, it is also scores (prints and symbols), books (theory) and information

about it (meta-data): virtually all the aspects, what can be found on Figure 1.2. If we limit ourselves to just one representation for a given problem, then the quality of the results is also limited by the quality of the data and natural limitations of certain representation. This may be one of the reasons behind the “glass ceiling effect” described by Downie [42], when one realizes, that despite the efforts and sophisticated solutions we cannot get close with the results of the algorithms to what humans with their music perception can easily achieve. Some music representations are simply lacking important aspects of the problem we are solving, for instance precise information about notes pitch, position and identification, which is necessary for search applications, is often scrambled, and usually lost, in audio stream, but it is precisely defined on symbolic level. On the other hand, timbral information is part of the audio signal but it is irrelevant on the symbolic level [85]. Just because of that, to analyze content at any level is equally important than analysis of this content at other layers (obviously, if the level contains relevant information to solve a given problem), regardless if this makes the problem easier or harder to solve. The ideal solution to any problem would involve analysis at all layers simultaneously, using integrated approach, which would automatically decide, which information are the most relevant for a given task as suggested by Vincent et al. [204] or by employing some mid-level representations that reduce the semantic gap between various levels, on which music is represented [125].

Another argument for symbolic music research is shared among those trans-disciplinary research fields, that join quite distant domains, like arts and sciences — it is quite unlikely for a researcher to acquire enough skills in both domains to effectively join them. This problem was identified by Downie [42], who looked into MIREX, which hosts evaluation of a number of MIR related tasks. He concluded, that the domain has biased towards audio-based techniques due to the fact, that most researchers have strengths in sound engineering which is closely related to computer science. He suggested that the results in some MIR domains would be better if more

musically meaningful features were analyzed and used, and symbolic representation is more musically meaningful than pure audio.

1.4 Research Questions

This dissertation addresses a number of research problems summarized in the questions below. The main focus is put on the development of the new techniques as well as adaptation of the existing ones from other fields, to aid analysis of symbolic music. Symbolic music for those problems is perceived as a sequence of symbols, and therefore it is approached in a similar way as a sequence of letters (i.e., text). The last two questions tackle the problem of structural segmentation, which is usually analyzed with audio data, while here we focus on using symbolic representation, showing the benefits of using it for this particular task.

How one can apply well-studied text processing techniques to symbolic music data? My current analysis of music data presented in this dissertation and other previous work shows that one can reasonably assume, that music in its symbolic form can be analyzed and used in the same way as text. Domains of computational music research and natural language processing deal with problems of similar nature using similar methods. I will expand this issue by analyzing how various methods of keyword extraction, or the choice of similarity measure affects performance of music-related AI tasks, like classification. The parallels of music and text will also be analyzed for enhancing proposed visualization approach to be more semantic, i.e., visualizing more high-level features as well as for developing some fundamental music operators, like similarity, based on their textual counterparts. Part of the answer to this question has been our submission that evaluated text-based similarity measures for MIREX 2011 challenge for symbolic music similarity [99].

How to represent and automatically evaluate melodies for music generation task within various frameworks of genetic algorithms? Genetic algorithms (GA) is a flexible tool that can be applied in various domains for many kinds of

data, so it is both interesting and important to determine how we can represent music for GA systems and which approach to genetic algorithms would be the most suitable for automatic music generation. Are there any specific features that make music applications in genetic algorithms unique? What is the fitness landscape of the domain? Is it diverse, or are we stuck in a single local optimum? Since running a genetic algorithm usually involves a large number of fitness evaluations of individuals in big populations, how one can approach automatic fitness evaluation to allow for wider applications of GA methods in music domain? Which GA framework works best for symbolically represented music?

How to visualize symbolic music such that both local patterns and high-level structure are easy to perceive? In order to get an idea about the content and the structure of a piece of music, the obvious solution is to listen to it, but this takes time — the same as the length of the piece. Is it possible to achieve a good perception of music content through visual matters without the need of spending time for listening? Does the proposed visualization method reveal structure, patterns, music textures and dependencies between voices or instruments? Can we make the visualization method suitable for various styles and genres?

How to automatically infer the structure of a symbolically encoded music piece? It has been shown that it is possible to create a system that detects boundaries between different parts of a piece that have different music features. Those systems frequently use audio data for analysis [147], but even then, it has been shown that extracting higher-level features from audio leads to much cleaner solutions [125]. This might be an indication that knowing the actual music events, or in general — symbolic music representation of a given piece, could lead to more accurate and more in-depth analysis of the content. It could be even possible that some dependencies that are not possible to be inferred from audio data will emerge by the proper use of symbolic data. Looking for such features is one of the goals of the thesis.

Does the process of finding a high-level structure benefit from high-level symbolic data, comparing to low-level audio? To answer this question, one can research and develop methods of detecting structure in music pieces based on the visualizations proposed for symbolic music. The cleaner, more precise symbolic data should allow for more accurate and precise piece partitioning; more high-level information could result in detection of related excerpts which would not look similar using current state-of-the-art techniques for audio data. The use of text parallels could aid detection of similar parts of the piece which could work better than the methods based on image processing used for structure analysis from audio similarity matrices. To answer those issues we will perform a comparison between our solution and a number of methods evaluated on audio data.

1.5 Contributions

Analysis of linguistic parallels in symbolic music research. We have introduced the history and levels of Natural Language Processing and aligned it with the equivalent concepts and theories in the field of Computational Musicology, commonly referred to as Music Information Retrieval. The similarities allowed for hypothesizing about the possible paths Computational Musicology can take. The findings from this analysis were presented during fMIR workshop held as a part of ISMIR 2010 conference, and published in conference proceedings [211].

Development of MIDI::Corpus module for handling corpora of MIDI files.

We have developed a PERL module to allow for simple handling and analysis of symbolic music corpora stored in MIDI files. It provides object-oriented interface, incorporates n-gram-based approach, and contains relevant analytics and statistics functions. It implements a number of similarity functions with bag-of-terms assumption, typically used in Data Mining, Information Retrieval or Natural Language Processing.

Collection and analysis of several classical symbolic music corpora. We have collected and analyzed several classical music corpora and compared its features with a pop music corpus (Beatles songs) and two text corpora: Wikipedia (English) and Orchid (Thai), seeking primarily for similarities and differences between music and written natural languages. The analysis covered areas from size statistics, entropy of symbols, distribution of terms (n-grams) and complexity.

Analysis and comparison of n-gram based approaches to composer classification and symbolic music similarity problems. Using n-grams and bag-of-terms assumption allows treating symbolic music in the same way as text. We have applied this philosophy in two tasks: authorship attribution (classification) and retrieval of similar items to the query. In the classification task, we have compared results with previous work as well as human judgements (Haydn/Mozart Quartet Quiz). The solution to the retrieval task has been submitted to 2011 MIREX Symbolic Melodic Similarity task [212, 213].

Design and application of a number of genetic algorithm models to melodies generation problem. We have presented how one can represent melodies in the framework of genetic algorithms and automatically evaluate their fitness based on a corpus of existing symbolic music pieces. This approach has been applied to various genetic algorithm frameworks, including generational, steady-state and Pareto optimization models. The main goal was to observe and understand how individuals behave under different models. The work has been presented on 2009 EvoStar conference [210].

Design and development of a new symbolic music visualization scheme based on self-similarity matrices. We have designed and developed a visualization system for symbolic music. It unveils both general structure and fine theme leading on a single image, through the concept of music self-similarity. The GUI

application has been developed, allowing anybody to use the system with their own MIDI files. The work has been presented during 2009 ICMC Conference [209].

Design and development of an automatic structure analysis algorithm for symbolic music. Since the structure of a piece was very apparent on the resulting visualizations, the natural extension was to derive the structure automatically. We have proposed a system, based on self-similarity matrices, that uses a modified Dijkstra algorithm to derive the piece structure. The system automatically labels obtained segments, indicating similar sections.

Quantitative comparison between algorithms for structural segmentation of audio and symbolic files on classical music pieces. We have compared the proposed segmentation system, operating on MIDI files, with five algorithms operating on audio data. We have used evaluation results obtained by an independent research group for audio data. We have manually aligned corresponding MIDI files with pieces in audio dataset and evaluated our algorithm under the same criteria. We have determined the statistical significance of the differences of performance of segmentation methods in question.

1.6 Publications

The following work have been published during the course of this doctorate research studies:

- Jacek Wołkowicz, Malcolm Heywood, and Vlado Kešelj. Evolving indirectly represented melodies with corpus-based fitness evaluation. In *EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, pages 603–608, Berlin, Heidelberg, April 2009. Springer-Verlag. [210]

- Jacek Wołkowicz, Stephen Brooks, and Vlado Kešelj. Midivis: Visualizing music structure via similarity matrices. In *Proceedings of International Computer Music Conference ICMC2009*, pages 53–56, August 2009. [209]
- Jacek Wołkowicz and Vlado Kešelj. Predicting development of research in music based on parallels with natural language processing. In *2010 Int. Society for Music Information Retrieval Conf.(ISMIR), fMIR Workshop*, pages 665–667, 2010. [211]
- Jacek Wołkowicz and Vlado Kešelj. Text information retrieval approach to music information retrieval. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2011. [212]
- Jacek Wołkowicz and Vlado Kešelj. Analysis of important factors for measuring similarity of symbolic music using n-gram-based, bag-of-words approach. In Leila Kosseim and Diana Inkpen, editors, *Advances in Artificial Intelligence*, volume 7310 of *Lecture Notes in Computer Science*, pages 230–241. Springer Berlin / Heidelberg, 2012. [213]
- Jacek Wołkowicz and Vlado Kešelj. Evaluation of n-gram-based classification approaches on classical music corpora. In *Proceedings to The Fourth International Conference on Mathematics and Computation in Music*, Montreal, Canada, June 2013. Springer-Verlag: Berlin, Heidelberg. [214].

The following publication is related to the research conducted in this thesis, and it has been published during the course of my doctorate studies, but it is a follow-up to my previous master’s thesis research:

- Jacek Wołkowicz, Zbigniew Kulka, and Vlado Kešelj. N-gram-based approach to composer recognition. *Archives of Acoustics*, 33(2008)(1):43–55, Jan 2008. [215]

1.7 Overview of the Dissertation

Chapter 2 contains the description of the digital data formats, in which music is available: audio and symbolic. It is followed by the description of the ways basic music features - pitch and rhythm - are encoded in symbolic music, followed by the analysis of the general approaches to symbolic music analysis. Chapter 2 concludes with analysis of linguistic parallels in symbolic music research.

Chapter 3 introduces n-gram based approach to monophonic symbolic music, used throughout the dissertation. It is then followed by the description of collected corpora of classical symbolic music along with a Beatles corpus, and two text corpora, used for comparison. Then, `MIDI::Corpus` module, used to manipulate corpora of MIDI files is introduced. This is followed by statistical analysis of collected corpora and application of two typical data mining tasks - document classification and retrieval of relevant documents to a given query. The latter constitutes our submission to 2011 MIREX symbolic melodic similarity task.

Chapter 4 contains description of our solution to melodies generation task with the use of Genetic Algorithms (GA). We propose representation of melodies which allows for the use of standard GA operators, and automatic fitness evaluation method based on a corpus of man-made music pieces. Several GA schemes were evaluated, which include Generational GA, Steady-State GA and multi-objective optimization. The chapter ends with a discussion on issues and possible extensions of the system.

Chapter 5 introduces a method of visualizing symbolic music pieces based on creating self-similarity matrices. The overview of the existing approaches continues with discussion about certain challenges and features of the system, including dealing with multiple tracks, defining a simple similarity function and applying filtering based on certain theme. It is accompanied with a number of examples, showing each step of the process. It follows with the description of the software package developed to allow for interactive creation and playback of our visualizations. The chapter concludes with more examples and future work.

Chapter 6 starts with the description of the existing approaches to the problem of structural segmentation. The methodology section contains redefinition of self-similarity matrix focused on structural segmentation, rather than on visualization. It is followed by exhaustive analysis of a proper similarity function that would give the most desirable base for a good structural analysis. It is accompanied with a number of examples, showing how well each similarity function unveils the structure of a sample piece. The chapter continues with the description of the method of deriving structural information from similarity matrices, from converting it to a single value (monochrome) layer, through definition of a novelty function, description of the modified Dijkstra algorithm that allows for jumps that indicate section boundaries, to the procedure of inferring final section boundaries and section labels. It is then followed with a description of the evaluation procedure, where our method is compared against five algorithms which were previously evaluated independently on a small testbed consisting of a number of classical music pieces. Chapter concludes with a comparison of the results with the analysis of significance of the obtained difference in performance of the algorithms.

Chapter 2

Background

A very important aspect of music is its polychotomy. From the digital perspective, music is usually represented in two forms: audio and symbolic, which are two, very different approaches to digitizing music. Currently, there is no good method of converting audio to symbolic representation, while the opposite conversion is, by nature, inaccurate. This sets the two worlds far apart. However, the research done in both areas — audio and symbolic music — deals, in fact, with the same object, a music piece. Both representations are two sides of the same coin: symbolic information and its audio realization.

2.1 Recorded Audio

Audio represents recorded music as a physical analog signal that is further quantified and discretized. Good quality output requires much storage space as quantization and discretization generates lots of data. To estimate those requirements, one can follow this simple analysis. Humans can perceive audio signals with frequencies up to 20kHz. According to the Nyquist sampling theory [138], the appropriate sampling frequency should double the maximal component of the signal, which in this case gives a requirement for sampling with at least 40kHz. The signal has to be also quantized with an appropriate level of details. Human ear is most sensitive in the range of 1-3kHz where its dynamic range reaches 120dB (between the absolute threshold of hearing and the threshold of pain). This is, however, much more than perceived dynamics of speech (30dB) or music (55dB) [208]. To keep 120 dB of dynamic range, assuming 6dB/bit, it is necessary to have 20 bits dynamic or 2^{20} discrete sound levels. On this psychoacoustic basis, standards were defined with the most popular,

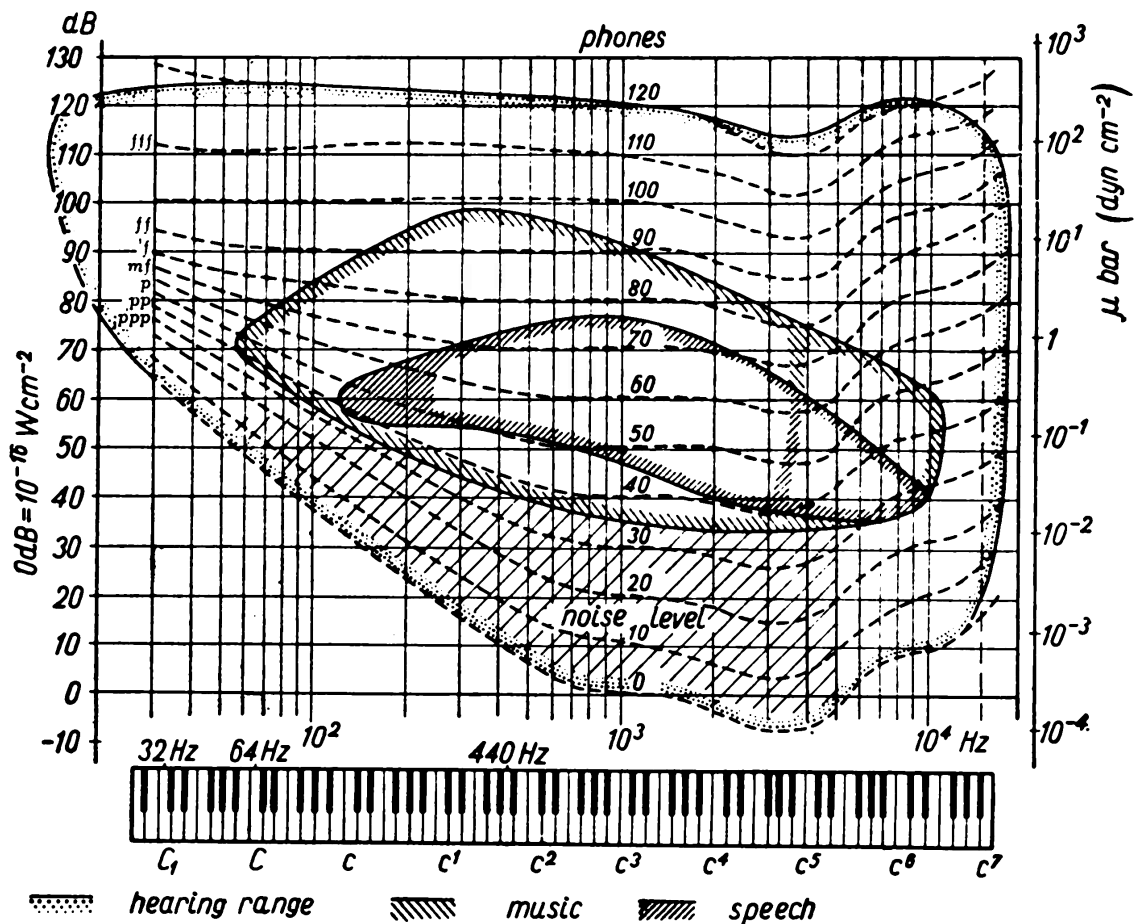


Figure 2.1: Human hearing range spans a range of frequencies and sound levels. Adapted from F. Winckel [208].

CD quality, being 44.1 kHz sampling frequency with 16 bits quantization, which captures most of the range of what humans can hear [55]. This gives the estimated requirements for bandwidth of a stereo Hi-Fi (High Fidelity) signal at the level of about 172kB per second or 10MB per minute or 600MB per hour.

To cope with the problem of audio file sizes, perceptual audio coding has been developed which allows on focusing only for those parts of the signal spectrum that are perceived by human ear. Those methods use various psychoacoustic phenomena, like temporal and frequency masking or hearing limits to achieve similar sound perception with much lower bandwidth required. Auditory tests show that current audio codecs are capable to deliver high-quality audio with 128kbps (or 16kB per second or 1 MB

per minute) limit on bandwidth [173]. This procedure reduces significantly size of the output files, however it keeps them in the same order of magnitude.

The principal modus operandi behind compression techniques is to deal with spectral representation of an audio signal, which is then used to extract features from the data. Most approaches to many tasks in the audio domain incorporate this approach. Typical workflow (from Pohle [156]) involves applying short and overlapping windows on the input signal to isolate frames, and computing certain features in each frame or global features for the entire piece. Feature extraction methods typically involve features belonging to the following groups (mainly from Tzanetakis [192]):

- Time domain features (based on signal): zero crossing rate, audio power.
- Timbral texture features (based on spectrum): spectral centroid, roll-off, flux, spread and flatness, as well as mel-frequency cepstral coefficients (MFCC), low-energy ratio.
- Rhythmic content features (based on beat histogram): amplitude of highest peaks, peak ratio, bpm (beats per minute) values of highest peaks, beat strength.
- Pitch content features (based on pitch histogram): most dominant pitch amplitude and octave range, main pitch class, main tonal interval relation, pitch detection strength and chroma vector (folded pitch histogram).

The obtained feature vectors are then subjected to a relevant data mining algorithm for a given problem.

2.2 Symbolic Representations

The second approach to storing music pieces on digital media is to encode music scripts prepared by composers, which define how to play them back to achieve the final effect as intended by the author. Music, as well as text, has its symbolic representation. Moreover, both music and writing are the only old human cognitive activities where symbolic representation is typically used. Others, like painting, sculpture, or dance did not have such symbolic notation.

The first text script system, cuneiform, was found by Sumerians in Mesopotamia about 3200 b.c. [182], while the first known inscription that may be treated as a basic music notation is dated back to 2000 b.c. [166]. Current notation used by musicians these days, known as Western Notation or Common Music Notation (CMN), originated in 14th century in France, but it is not a definite distinction as it had been evolving constantly since early 11th century (Guido d'Arezzo) to converge in 18th century, with Bach's introduction of 'equal temperament' [60].

According to Isaacson [85], this slow evolution led to creation of certain metaphors that reflect and define our music understanding. We perceive some note as *higher* and *longer* than others which reflect their position and space, they occupy. Melodies *ascend* and *descend* creating a sense of direction. Motifs are *stretched* and *compressed*, reflecting their appearance on the score [85].

Most music events in Common Music Notation are notes positioned on a five-line staff for accurate pitch definition in the twelve-tone chromatic scale. The look of each note defines its length in comparison to other notes. For more complex scores, a staff system is used, that comprises many staves. It is usually one staff or one staff group for an instrument.

The use of additional marking is required for precise note placement and performance guidelines:

- Rests mark periods of silence in music.
- Grace notes, trills and other ornament marking may introduce additional notes not stated explicitly in the score.
- Clefs, key signatures and accidentals may alter a note's pitch. The use of them is necessary to display all possible pitches in a twelve-tone chromatic scale since pure five-line staff accommodates only diatonic notes in C key.
- Tempo marking such as dots, ties, fermatae, some verbal indications, affects duration of notes and rests.
- Articulation marking affects the ways a note is to be played.

Name	extension	format	description
Finale	.mus	proprietary/binary	—
Sibelius	.sib	proprietary/binary	—
MusicXML	.mxl	open/textual	XML format
LilyPond	.ly	open/textual	L ^A T _E X style
Guido	.gmn	open/textual	similar to LilyPond, compact
Humdrum	.hrm, .krn	open/textual	pseudo tabular, research oriented
MIDI	.mid	open/binary	encodes only audible features

Table 2.1: Score formats.

- There are more, mainly verbal, marking that define music expression.

Music symbolic representation can not be directly ported into computers like text is, but there are ways to convert this representation to be more computer friendly. Their aim is not to store the exact and accurate information about the recorded sound but to digitally encode music score, which can be later used by a musician to learn and perform the piece. This approach keeps just the information intended by the composer, discarding all the details about actual recording.

Symbolic representations are also used by computer notation software, whose aim is to capture all possible visual features of a score, so that one can print it, without losing any important information from the original score. All scorewriting systems allow for playback of an edited file, which proves that those formats are capable of storing symbolic music information (contrary to scanned versions of the same score). There are two leading commercial scorewriting programs (Finale [84] and Sibelius [83]), and both of them use their own proprietary binary file format (.mus and .sib respectively). There is also a number of open standards, and their transparency makes them more appealing for keeping symbolic music for research and analysis purposes. Some of them, e.g., Humdrum [82], are designed specifically to meet requirements of music researchers, but for most of them, the main purpose is to serve music scripting. The most popular formats for storing symbolically represented music are described in Table 2.1.

2.2.1 MIDI Protocol

MIDI represents a different approach. Instead of storing actual audio information, which requires lots of space, it keeps only information about music events such as pressing or releasing a key on some predefined instrument. Unlike scorewriters, it does not intentionally store any information necessary to print the actual music score, only audible events are stored in a MIDI file, however approximate printing is still possible.

MIDI stands for Musical Instrument Digital Interface and it was designed as a communication protocol to control and synchronize musical instruments, computers and other electronic equipment. The aim was to extend a set of event messages beyond media signals such as pitch and intensity of a played note, to other types events like control signal for various audio and non-audio features and precise clock signals for synchronization. MIDI standard defines not only the electronic protocol, but extends to hardware specification as well (RFC 6295 [104]). MIDI standard has been adopted widely in the industry and remained unchanged since its introduction in 1983 despite all the changes in the technology.

Standard MIDI File (SMF) is a container for MIDI protocol messages. It contains concurrent channels and tracks. Each channel is a container for events, called MIDI messages. There are three categories of these messages:

1. Channel (voice) messages — represent media type, playback events, like Note-On (note initiation), Note-Off (note termination), Key Pressure (sound volume), etc.
2. System (real-time) messages — messages controlling real-time events that may happen during playback, such as Clock, Tick, Start, Stop, Continue, Reset. They affect control and flow of other sync messages.
3. System Common messages — provide additional information, which is not essential for playback, such as System Exclusive messages (mostly textual data) and playlist controllers (Song Select, Song Request).

Both channel and real-time messages depend on time. Time in Standard Midi File is counted in microseconds, so any event can be precisely located. While time information remains more or less continuous (as it is in audio data), pitch information is encoded symbolically. Instead of frequency value, a key number on a virtual instrument is used. There are 128 possible notes on a MIDI device, numbered from 0 to 127. The middle C (261 Hz for most temperaments) is note number 60, and, as in the well-tempered system, the frequency of each note is $2^{1/12}$ times larger than the previous note. This solution puts a constraint to the use of MIDI for western music's chromatic, twelve-tone scale. The use for other systems is problematic, although possible with the use of pitch bend events, if they do not fit easily into twelve-tone well-tempered scale.

These features put MIDI somewhere in between audio and music score formats. Like audio, it operates in continuous time, keeping information on audible events. Except for fixed notes frequency levels, information in MIDI files is unstructured and unbounded by rules of music. High-level music features like phrase structures, key, rhythm are not part of MIDI standard, although they may be stored as metadata in Sys-Ex messages. However, in most of the cases it is not present and scorewriting software has to infer those information from bare notes. Along with audio data, its main purpose is for recording and playing back music performances. With a proper synthesizing software and well-formed data file, the idea behind MIDI is that it can produce quite good results in terms of perception assuming that it stores a tiny fraction of the original audio information (typically less than 0.1% comparing to Hi-Fi audio), still being able to recover most of the score information. In most cases this is enough for research where one does not deal with very high-level and abstract music features. Other positive aspect of using MIDI in research is that it is, and has been, used widely, therefore it is quite easy to acquire a corpus of any genre of music, large enough to perform statistically significant analysis on it. It is usually not the case with other scripting formats.

MIDI time. MIDI specification has a very intricate way of defining time. The reason is to keep the definition of each measured note the same regardless of local tempo fluctuations. MIDI file uses *ticks* as internal time dimension for all the events. The way, how ticks map to real life time, i.e., seconds, is defined with two parameters:

- **division** appears once, in the header and defines how many ticks fall into one quarter note (**tpqn**), typically a value of 96 is used, but anybody can set it to a higher value if a more fine grained representation is required.
- **tempo** events are meta events scattered across a MIDI file, each defining how many microseconds lasts a quarter note. Their position, as other MIDI events, is measured in ticks. Tempo events define a tempo map, indicating how the speed of music fluctuates with in a piece. The default tempo assumed at the beginning of a file is 500000 microseconds per quarter note, or 120 beats per minute.

For example, to calculate the duration of a piece, a following formula can be used:

$$duration = \frac{0.5p_1}{d} + \sum_{i=1}^{N-1} \left(\frac{t_i (p_{i+1} - p_i)}{1000000d} \right) + \frac{t_N (p_{max} - p_N)}{1000000d} \quad (2.1)$$

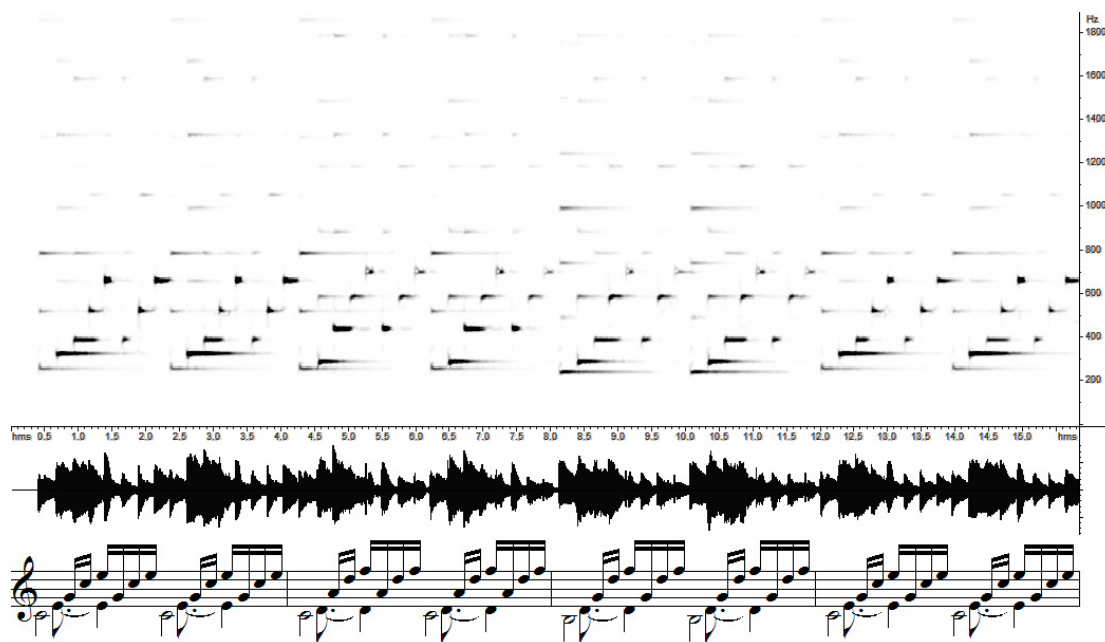
where d is the division from header, N is the number of tempo events in the file, p_i is a position on a ticks scale of i th tempo event, t_i is the tempo value defined by i th tempo event, and p_{max} is the position of the last note-off event in the file.

2.3 Format Conversions

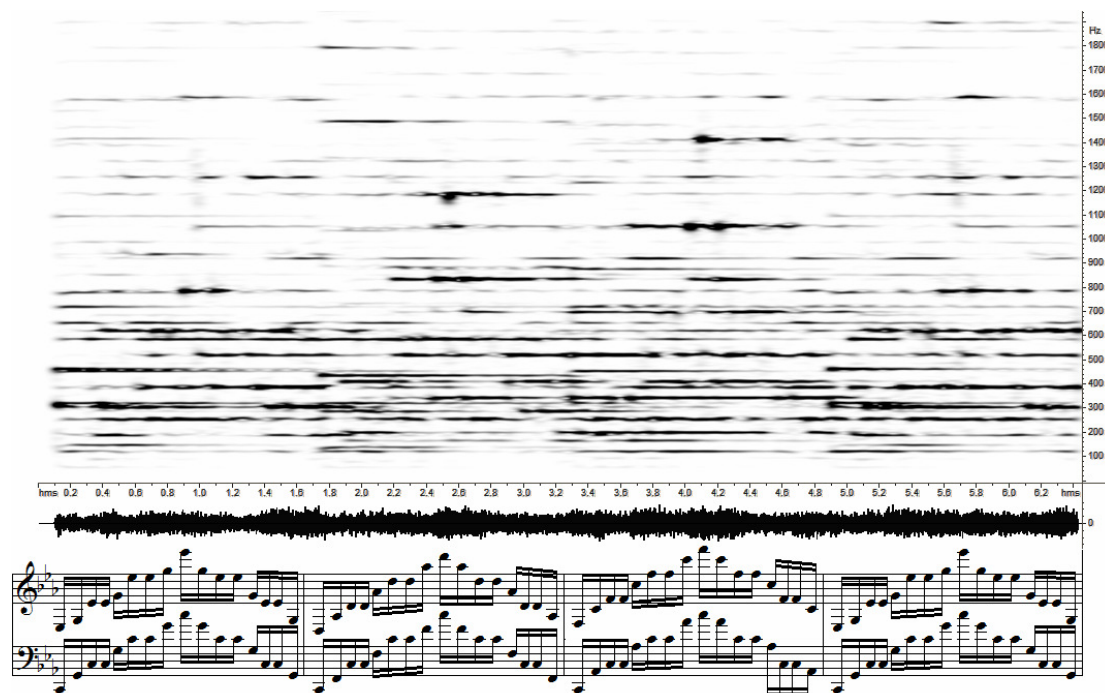
As it has been pointed out in the previous sections, music digital representations span various levels and the most important distinction is between three of them: audio (digital signals), music events (e.g., MIDI, unstructured audio event stream) and music score (music notation that encodes all high-level music features). Tasks at each of these levels experience different issues and problems so it is important to realize what are the limitations of conversions between those formats. In practise, there are four steps to be considered, two top-down (score to events, events to audio)

and two bottom-up (audio to events, events to score). Both top-down steps are quite straightforward, especially score to event representation, which requires only identification of audible features in the score with all the features that affect them, and decoding them into events (MIDI). Music events-to-audio process, which is called synthesizing, involves either synthesis of artificial sounds from a computational model or by using a set of digital audio samples, which is similar to the process of using fonts in word processing software. Both of these methods work quite well, although the final result may be different from the original recording. It is not surprising, given the fact that MIDI format can be considered as a lossy compression with a ratio of more than 1 to 1000. Nevertheless, music is primarily an art, so a lot depends on the performers interpretation and subtle details, that define player's uniqueness and artistry. This cannot and should not ever be captured and accurately re-generated using any formal symbolism, since true art should be inimitable.

The problem starts when one wants to transcribe pieces from audio to symbolic representation. The transcription from MIDI to score format poses interpretation problems, because raw music events information does not have any high-level or structural components. This information has to be implied from the event sequence, which is not always obvious, however with some feedback from the user it can be done well. The real problem is with note recognition from raw audio data. It is a complex process and it is effective usually for only simple, monophonic pieces. Recent approaches to this problem focus on certain aspects of transcription, like main melody, bass line, harmony or rhythm tracking, being able to correctly assign from about a half to 75% of the events [43, 164]. Surprisingly, a trained professional or a music school student should be able to recognize and write down with relative ease even a complex polyphonic piece. The problem lies in the fact that in recorded audio all events are blended together to form a single sound, and most of the notes that conform this sound usually span most of the spectrum. From the computational analysis perspective, note and source separation is a very complex process while it happens to be a relatively easy procedure for human brain.



(a) J. S. Bach, Prelude C major from WTK 1 (BWV 846)



(b) F. Chopin. Etude C minor op. 25 no. 12

Figure 2.2: Score samples with its corresponding spectral images.

To illustrate the problem of audio transcription, Figure 2.2 contains two excerpts of piano music, one is a simple monophonic example (only one note is being played at a time) while the second contains passages of notes in rapid succession in both hands simultaneously. Both excerpts are accompanied with spectral images that correspond to each of those excerpts and are aligned accordingly. Since spectral (and cepstral) images of the actual audio serve as a base for transcription analysis, it is clear that although in the first case it would be relatively easy to recognize component notes, the second example poses serious problems for this analysis. It results from the fact that the spectral image of a note played on any instrument is represented in its spectral image not only by a single frequency peak, but it is always accompanied by a series of harmonics that can mix-up with other notes and other note's harmonics and bands of noise that can mask components of other notes played at the same time. The resulting spectral image is a sum of all the images of all notes played at the same time, and in most cases it is also blended with some background noise. The result is mathematically not easy to separate which leaves researchers in this area with complicated interpretation problems.

There is, however, a method to avoid the need of doing transcription for tasks, that requires audio along with its precise symbolic representation. This idea comes from Natural Language Processing, and deals with automatic machine translation. It turned out, that creating vocabularies and building a linguistic model to translate from one language to another is a very complicated task, that brings quite poor and sometimes surprising results. Anecdotal is a translation of a passage from the Bible where "The spirit is willing, but the flesh is weak" had been translated back and forth from English to Russian and turned into "The vodka is good, but the meat is rotten". The way to overcome this problem is, instead of building complicated language models, analyze plenty of bi-lingual texts, align them and extract patterns, perfectly translated from one language to the other. Then for any input text, try to compile it from the list of known phrases and their translations, to form a translated

text. It turns out that this approach outperforms significantly classical machine translation systems [18].

Audio transcription task resembles machine translation from one “language” — audio into another — score. Instead of creating complex models, what if one finds for a given audio file a corresponding MIDI file, then the only thing missing for the transcription would be to find the alignment, i.e., mapping from audio time to MIDI time, between them. Unlike transcription, alignment can be done accurately [31], which allows application of symbolic-based methods to problems, where typically audio is the only available source of data.

2.4 Monophonic Music Encoding

Monophony is additional requirement for symbolic music, which enforces that each voice has only one note sounding at any time. This constrains music data not to have chords nor parallel voicing. For the purpose of applying bag-of-words approach, used primarily in this thesis, monophonicity have to be enforced for each voice separately, which allows for multiple concurrent voices with notes played at the same time. If these requirements are not met, then one deals with *polyphony*. Monophonic music has the following benefits over polyphonic music:

1. Since one can define order of notes, as they follow one another, it is possible to introduce relative measures between them. This allows to satisfy tempo and transposition invariance, which is one of the primary requirements in symbolic music analysis.
2. Complexity of the linear (string) models drop significantly, as one does not have to deal with the lattice of all possible connections between notes, which grows exponentially if one allows for chords to appear [44].
3. It makes music similar in nature to written texts, where letters and words follow one another and it is always clear which is *before* and *after*, so adaptation of

text-based methods (either using string or bag-of-words approaches) is relatively straightforward.

It is possible to reduce polyphonic music either through splitting polyphonic channels to a number of monophonic ones [44], or by dropping the most irrelevant notes to preserve the most of melody perception, while making it monophonic (e.g., through skyline method [193]).

The term *polyphony* is also used in music theory in different context, that may introduce confusion in certain areas. A piece is *polyphonic (music theory)* if all the voices are equally important and lead their own melody. This is opposite to *homophony (music theory)*, where one voice has a distinct lead over all other, subordinate voices. The equivalent of *monophony* in computational musicology is usually called *monody* in music theory. As a result, from music research perspective all *homophonic (music theory)* pieces are *polyphonic (computational musicology)* as they feature many concurrent voices, but not *polyphonic (music theory)*, not all of them are equally important. Similarly, *polyphonic (music theory)* pieces, like Bach’s fugues, are usually *monophonic* for computational models, as each voice features a single melodic line, so it is not *polyphonic (computational musicology)*. The actual meaning of these terms depend on context and domain, which is usually easy to recognize.

Notes in a monophonic melody represent actual music events, placed in time (rhythm) and frequency (pitch) space. There are several approaches to computing basic features from both dimensions and they are summarized in the following sections. They are primarily grouped into absolute, semi-relative and relative, depending whether their values are taken from a single note or the relations they have with other notes. Although, here we deal with pitch and rhythm separately, they are often combined to create a single pitch/rhythm feature space [136] or intertwined [33] to get much more complete representation.

2.4.1 Pitch Encodings

Absolute

Absolute notes encoding places them directly and precisely on a pitch scale, therefore they are useful for scorewriting or playback purposes:

1. **Frequency** maps a note n to a real positive number representing typically the base frequency or the fundamental tone of a note $f(n)$. It is historically assumed that $440Hz$ represents *the middle A*. Since it is a real scale — it can represent any sound, even outside of western tonal music system. It is typically assumed that humans can hear sounds in the range from $20Hz$ to $20kHz$.
2. **MIDI pitch** maps a note to MIDI event' pitch number, an integer from a range $[0, 128]$, however theoretically any integer number can be used, as 0 and 128 limits are arbitrary. To map from MIDI pitch to frequency, one has to define a meeting point for frequency and pitch scale, and since MIDI standard defines that *the middle A* is 69, so $f(n) = 440Hz \iff p(n) = 69$. Assuming equal temperament, every MIDI pitch can be mapped to the respective frequency using the following formula:

$$f(n) = 440Hz \times 2^{\frac{p(n)-69}{12}} \quad (2.2)$$

or vice versa:

$$p(n) = 12 \times \log_2 \frac{f(n)}{440Hz} + 69 \quad (2.3)$$

with the following two features, that the frequency ratio between two consecutive pitches is constant:

$$f(n_1) = 2^{1/12} \times f(n_2) \iff p(n_1) = 1 + p(n_2) \quad (2.4)$$

and that the frequency doubles every octave (12 notes):

$$f(n_1) = 2 \times f(n_2) \iff p(n_1) = 12 + p(n_2) \quad (2.5)$$

There are other temperaments used historically, like Pythagorean (based on circle of fifths and stacking $3/2$ frequency ratios between them) or Natural

(based on harmonic series, where frequencies are tuned to represent fractions in a form $\frac{k+1}{k}$ where $k > 0$ for certain intervals). They do not satisfy condition from Equation 2.4, but they are all constructed to satisfy Equation 2.5. Those historic temperaments are still important as many instruments (most wind instruments) or playing techniques (e.g., string flageolet) generate tones in natural scale.

3. **CMN (Common Music Notation) pitch** is used for music notation purposes and precisely defines the position and look of each note on the score and its harmonic function within a given key. Rizo [163] defines a note as a 3-tuple, $n \stackrel{def}{=} \langle d, a, o \rangle$, where:

- $d \in \mathcal{D}$, and \mathcal{D} is a diatonic note set, such that $\mathcal{D} = \{C, D, E, F, G, A, B\}$,
- $a \in \mathcal{A}$, and \mathcal{A} contain accidentals, where $\mathcal{A} = \{\flat\flat, \flat, \natural, \sharp, \times\}$,
- $o \in \mathcal{O}$, and \mathcal{O} defines octaves and $\mathcal{O} = [-1, 9]$, but again, lower and upper bounds are set arbitrary, so theoretically $\mathcal{O} = \mathbb{Z}$.

If one assigns specific values to \mathcal{D} and \mathcal{A} set elements, namely interval from C in semitones ($C \stackrel{def}{=} 0$, $D \stackrel{def}{=} 2$, $E \stackrel{def}{=} 4$, $F \stackrel{def}{=} 5$, $G \stackrel{def}{=} 7$, $A \stackrel{def}{=} 9$, $B \stackrel{def}{=} 11$) and accidentals to the original pitch modification they carry ($\flat\flat \stackrel{def}{=} -2$, $\flat \stackrel{def}{=} 1$, $\natural \stackrel{def}{=} 0$, $\sharp \stackrel{def}{=} 1$, $\times \stackrel{def}{=} 2$), then the mapping from Common Music Notation to MIDI pitch can be defined as:

$$p(\langle d, a, o \rangle) = 12(o + 1) + d + a \quad (2.6)$$

Since this notation is more musically complete than MIDI pitch, as it defines also the role of a note for a given key, it is redundant with regards to defining pitch, for example $p(\langle C, \sharp, 4 \rangle) = p(\langle D, \flat, 4 \rangle)$, but $\langle C, \sharp, 4 \rangle \neq \langle D, \flat, 4 \rangle$.

Semi-relative

Semi-relative mappings calculate note's pitch in relation to static, or locally static, reference, which is usually either C note, or local tonic, assuming we either know or have inferred the key of a melody. They partially satisfy transposition invariance

requirement, because if the same melody is transposed into two different keys, knowing them allows us to identify their identity. However, this approach assumes, that we know the correct key of the melody, which might not be the case, and it is insensitive to local modulations or progressions. The latter might not be necessarily a problem, if we assume that a modulation or a progression defines a different melody, which is fair to do if that is what is required.

The relation to particular reference note loops the values for each octave, creating an equivalent of mathematical 'modulo' operation. According to Rizo [163], those modulo- n , or base- n representations may have two useful properties:

- accommodation of enharmonic pitches — two notes with the same MIDI pitch and different CMN pitch (e.g., $\langle E, \sharp, 3 \rangle$ and $\langle D, \flat, 3 \rangle$) should have two different codes to accommodate CMN notation.
- interval invariance — same interval (i.e., numerical difference of MIDI pitches) between any two notes should lead to the same numerical value, modulo n .

There are four approaches to semi-relative coding:

1. **diatonic encoding** divides an octave into 7 different equivalence classes, taken from the first dimension of CMN pitch notation, i.e.:

$$\text{base7}(\langle d, a, o \rangle) = d \tag{2.7}$$

Typically for analysis, number from 0 to 6 (or from 1 to 7, as in Figure 2.3) are used, but historically, diatonic tones were encoded using syllables: *do*, *re*, *mi*, *fa*, *sol*, *la* and *si(ti)*. Discarding accidental information leads to generalization of a melody, and it does not hold any of base- n desired properties.

2. **chromatic encoding** divides an octave into 12 equivalence classes, commonly known as pitch classes, each corresponding to a semitone of the octave as defined by equal temperament. It does not accommodate for enharmonic pitches, but preserves interval invariance. Since the first property is relevant mainly for

notation purposes, this encoding is frequently used for analysis purposes (e.g., in [128]). It can be directly derived from MIDI pitch value:

$$base12(n) = 1 + (p(n) - p(r)) \bmod 12 \quad (2.8)$$

Where r is a reference note, usually $\langle C, \natural, 4 \rangle$.

3. **base21 encoding** has been suggested to “provide a sufficient number of numerals to differentiate each enharmonic tone within the range of single sharps and flats” [165]. It accommodates therefore for representing enharmonic pitches, but it is not interval invariant, e.g., two intervals (here minor thirds) have the same difference in pitches, $p(\langle E, \flat, 4 \rangle) - p(\langle C, \natural, 4 \rangle) = 3$ and $p(\langle G, \flat, 4 \rangle) - p(\langle E, \flat, 4 \rangle) = 3$, but different base21 values ($base21(\langle E, \flat, 4 \rangle) - base21(\langle C, \natural, 4 \rangle) = 5$ and $base21(\langle G, \flat, 4 \rangle) - base21(\langle E, \flat, 4 \rangle) = 6$) [163]. Figure 2.3 shows, how base21 values are assigned to notes.
4. **base40 encoding** — it was suggested by Hewlett [74], to acquire interval invariance property, while accommodating for enharmonic pitches representation. Thus it can be used in various analysis tasks like *base12*, keeping information about note’s visual appearance. It has been used as intermediate pitch representation in Humdrum system [82]. One can see on Figure 2.3, how base40 values are assigned to notes.

Relative

Relative pitch values, calculated using a local reference point, which is one of the neighbouring notes, satisfy transposition invariance requirement. The same perceived melody, played in different keys would result in the same dependencies between each pair of notes, although absolute values may be different. Relative pitch features are based on the notion of interval — the distance in pitch (i.e., MIDI pitch), usually between two consecutive notes. From each sequence of N notes, there is $N - 1$ intervals, so depending on which neighbour we refer to (preceding or following) either the first

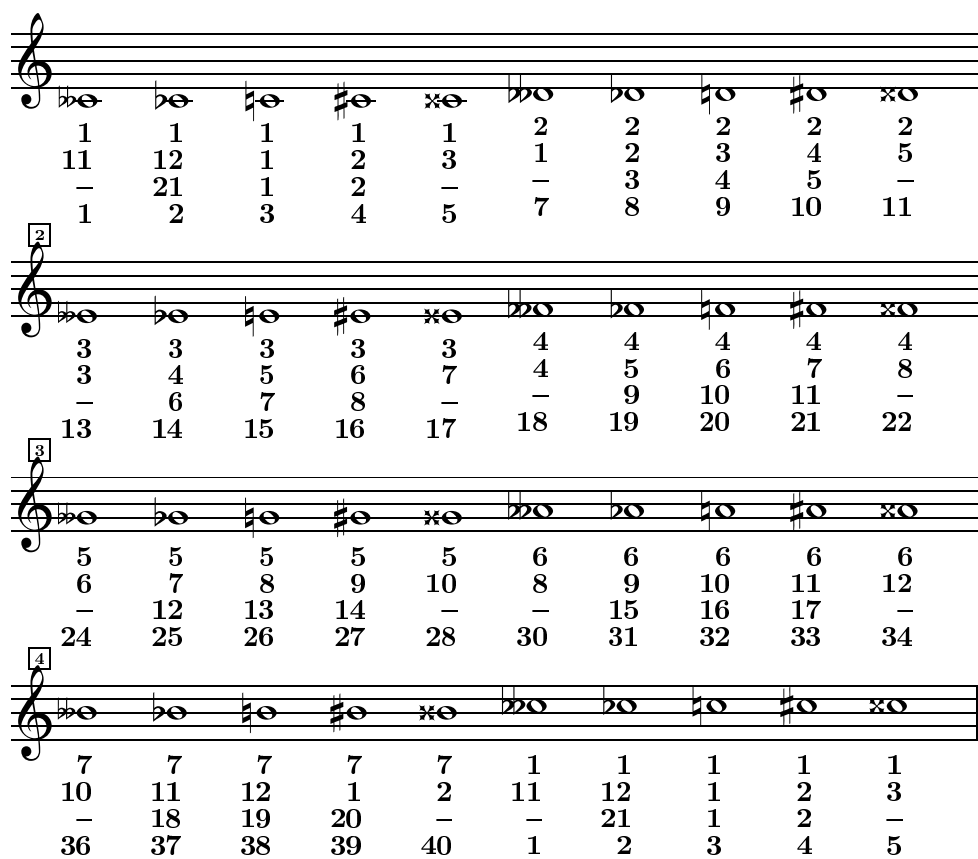


Figure 2.3: Pitch encodings for each note (from the top): diatonic, chromatic, base21 and base40.

or the last note does not have an associated interval. The choice of a reference point is not important as long as the consistency is preserved. Here, interval calculations will be conducted with regards to notes that follow a given note. One can classify interval types into several groups:

1. **Melodic Interval** is a difference in value of two consecutive MIDI pitches. This approach is very frequently used in the literature, e.g., [33, 128, 136, 144] as it allows for precise notes' pitch definition with the ability to restore the original sequence of notes, as long as the starting point is known. Melodic interval between two neighbouring notes can be computed directly from MIDI pitches, and expresses traditional music interval in semitones:

$$mi(n_i) = p(n_{i+1}) - p(n_i) \quad (2.9)$$

where $i \geq 1$ is a position of a note n_i in the input notes sequence.

2. **Clipped Melodic Interval**. Melodic intervals can be big, but large intervals are relatively rare. In order to limit the number of possible intervals, clipping to some maximum value can be used. Typically an octave (12 semitones) sets the limit [140], but any arbitrary value can be used. In case of an octave, the formula changes to the following:

$$cmi^{12}(n_i) = \max(-12, \min(12, p(n_{i+1}) - p(n_i))) \quad (2.10)$$

3. **Melodic Interval Classes** is another approach to limit the feature space, which is more musically meaningful. Intervals of up to 12 semitones (i.e., an octave) are called simple intervals, and above that — compound intervals [218]. For example, a major tenth (16 semitones) is composed of an octave (12 semitones) and a major third (4 semitones), and it is typically referred to as a third over the octave and it is perceived as such. It is, therefore, reasonable to implement this reasoning in a music analysis system [193, 196]. Unlike for clipped melodic intervals, only the octave (12 semitones) is a musically meaningful threshold, and the formula for melodic interval classes can be expressed

as follows:

$$mic(n_i, n_{i+1}) = \begin{cases} mi(n_i) & \text{if } |mi(n_i)| \leq 12 \\ mic(n_i, n_{i+1}^{8-}) & \text{if } mi(n_i) > 12 \\ mic(n_i, n_{i+1}^{8+}) & \text{if } mi(n_i) < -12 \end{cases} \quad (2.11)$$

where for any $n_j = \langle d, a, o \rangle$, $n_j^{8+} = \langle d, a, o + 1 \rangle$ and $n_j^{8-} = \langle d, a, o - 1 \rangle$.

4. **Pitch contour.** There are applications, mostly those dealing with user input, that have to be robust in terms of handling inaccuracies in the data. In many cases, music information retrieval systems using query by humming (QBH) input approach face a problem that users can not sing clearly. Often, the only correct information is the direction of the melody. To mitigate this problem, pitch contour is frequently applied. It encodes the direction of the melody, instead of a precise interval. Uitdenbogerd and Zobel suggested using three symbols, *U* for unison, *A* for ascending and *D* for descending melody [193,197]. Grachten et al. in their comparison of various approaches to melody coding for measuring similarity found a very strong correlation (0.95) between results using melodic intervals and pitch contour [67], which indicate, they might be used interchangeably. Pitch contour uses a three symbols alphabet and it is often expressed as:

$$pc(n_i) = \text{sgn}(mi(n_i)) \quad (2.12)$$

5. **Fine pitch contour.** There are situations when pitch contour's precision is too small, and using melodic intervals produces too few results. In that cases, increasing the number of levels for calculating the contour is a good trade-off between them. Uitdenbogerd and Yap have found, that users, especially those with intermediate music background, can successfully query a retrieval system that uses fine pitch contour, almost as well as those using normal pitch contour, and much better than systems that use strict melodic intervals [197]. Downie [38] analyzed entropies of obtained interval groups to find the optimal division points defining pitch classes. Muellensiefen and Frieler [132] referred

to the process of grouping intervals as *fuzzification*, suggesting nine levels, four ascending (i.e., seconds, thirds, fourths with fifths, and sixths and up), same four descending and unison. Berthelemy [6] classified intervals into those that join melody (seconds) and disjoin, creating jumps (thirds and up). In the experiments conducted in this thesis the following definition is used:

$$fpc(n_i) = \begin{cases} -2 & \text{if } mi(n_i) < -3 \\ -1 & \text{if } -3 \leq mi(n_i) \leq -1 \\ 0 & \text{if } mi(n_i) = 0 \\ +1 & \text{if } 1 \leq mi(n_i) \leq 3 \\ +2 & \text{if } mi(n_i) > 3 \end{cases} \quad (2.13)$$

2.4.2 Rhythm Encodings

Second and equally important dimension of a note is rhythm, which describes its temporal placement. One can treat rhythm in similar way as we deal with melodic aspects of notes, with a few differences. One of them is the existence of rests — breaks in melody, when one note’s offset time does not match and it is before next note’s onset time. In the literature, different approaches have been used, but since rests do not carry pitch information, they would break interval passages. In this work we just omit them, since some rhythm encodings, like IOI or IOR escape the needs of dealing with them at all.

Absolute

Absolute methods of representing rhythmic features give results in time domain, measured typically in seconds. Since they do not stand in any relation with each other, they do not hold tempo invariance property, which states, that the same melody played in two different tempos should be represented in similar way.

1. **onset time, duration.** Unlike melody, where frequency (or pitch) is the only basic physical feature, each rhythm event is defined with onset and offset time,

or more frequently, onset $o(n)$ and duration $d(n)$, which is a difference between note’s offset and onset time. It is usually expressed in seconds, however MIDI specification uses *ticks* measure. In a well-sequenced MIDI file, ticks are supposed to represent more semantically meaningful time, keeping the same ticks value for same notes regardless of local tempo fluctuations. The detailed description on how to convert from MIDI ticks to real time is presented in Section 2.2.1.

2. **Inter-onset intervals (IOI)** present a very similar concept to duration, but instead of relying on notes offset times to determine when a note ends, IOI assumes that the offset time of a note is the onset of the following note [68]. In this approach, we not only solve the ‘rests’ problem, where they are simply merged into the length of the preceding note, but also allow for notes to overlap, i.e., when the offset time of a note is after the onset time of the next note. As a result, IOI stream is always a continuous stream of strictly monophonic events, so in turn the process of obtaining IOIs acts like skyline monophonization method proposed by Uitdenbogerd and Zobel [193]. IOI function can be expressed as follows:

$$ioi(n_i) = o(n_{i+1}) - o(n_i) \quad (2.14)$$

This gives $N - 1$ values for a series of N notes. In general, the duration of the last note is often used as the last, N ’th, IOI.

Semi-relative

Semi-relative approaches for rhythm work similarly to pitch approaches, the only difference is that, since time space is continuous and pitch is discretized, a quantization step is often performed.

1. **duration classes.** Similarly to melody fuzzification, Muellensiefen and Frieler applied a fuzzification process to durations, classifying notes into 5 classes comparing to the most frequent duration in a given piece [132]. They came up with

the following classes:

$$dc(n) = \begin{cases} +2 & \text{if } \frac{d(n)}{d_{max}} > 3.3 \\ +1 & \text{if } 1.8 < \frac{d(n)}{d_{max}} \leq 3.3 \\ 0 & \text{if } 0.9 < \frac{d(n)}{d_{max}} \leq 1.8 \\ -1 & \text{if } 0.45 < \frac{d(n)}{d_{max}} \leq 0.9 \\ -2 & \text{if } \frac{d(n)}{d_{max}} \leq 0.45 \end{cases} \quad (2.15)$$

where d_{max} is the most frequently occurring duration.

2. **note symbols.** Typically in music, the duration is expressed using different graphical symbols for expressing how long each note should last. What speed the piece is played (what is the actual duration of each note is suggested by the composer in the tempo description at the beginning of a piece) is left up to the performer. Those symbols define duration with reference to the whole note, and they are named after what fraction of a whole note they represent (whole note, half note, quarter, eighth, etc.). This representation is widely adopted by musicians, and used in scorewriting systems.

In computer music research, note symbols are usually referred to as *relative to a standard duration representation* [163, 165, 197], and can be expressed with the following equation:

$$dc(n) = \frac{d(n)}{d_{ref}} \quad (2.16)$$

Typically, the end result is quantized and rounded to some power of 2, which gives the precision of, for example, eighths (i.e., 2^{-3}) or sixteenths (i.e., 2^{-4}), or it might be left as a real value. This procedure may be also applied to IOI values resulting in quantized IOIs [136].

Relative

Relative approaches to rhythm encoding work similarly to relative pitch encoding, with several important differences. Pitch is already quantized and since usually one

needs discreet classes, rhythmic features need to be quantized separately. Secondly, as human perception of pitch grows exponentially with frequency, the same can be observed on rhythm, which lacks this fundamental logarithmic representation that MIDI pitch and all its derivatives feature. The last difference lies in the fact, that note's duration can be calculated either to its nominal offset time, or until the beginning of the next note (IOI), as it was discussed earlier. Typically, relative feature functions use IOI, but absolute note durations may be used as well. Similarly to melodic relative methods, this approach allows to achieve tempo invariance, so two equivalent melodies played in different tempos, will be represented using same, relative features.

1. **Inter-onset interval ratios (IOR)**, sometimes abbreviated differently in the literature, represent the ratios between two durations (or rather IOIs) of two consecutive notes:

$$ior^r(n_i) = \frac{ioi(n_{i+1})}{ioi(n_i)} = \frac{o(n_{i+2}) - o(n_{i+1})}{o(n_{i+1}) - o(n_i)} \quad (2.17)$$

Since three notes are involved in the calculation of a single feature (i 'th, the preceding $i - 1$ 'th and the following $i + 1$ 'th), N notes would give us $N - 1$ melodic interval features and $N - 2$ IOR features represented in rational numbers, hence the r index. Doraisamy [33] proposed intertwining *melodic* and *rhythmic* features, in the pattern like *mrmrmm*, which allows for one less rhythmic feature in each string. She also used a binning approach to quantization, where named bins, derived from the data distribution, were used to classify IORs into categories. Pardo and Birmingham [144] used linearly distributed bins on the logarithms of IOR, and aimed to preserve sufficient information to discriminate between melodies, while keeping the alphabet small. They concluded that even a very small number of bins (4) allows for clear distinction between different melodies although the information needed to recover those durations is certainly lost at this point. Meek and Birmingham [128], looked into this issue and suggested using 4 bins per each exponent of 2 in the ior^r space. The definition I propose results in integer numbers each representing a bin of $1/p$ of

each doubling of IOI time:

$$ior(n_i) = \lfloor p \times \log_2 ior_{n_i}^r + 1/2 \rfloor = \lfloor p \times \log_2 \frac{o(n_{i+2}) - o(n_{i+1})}{o(n_{i+1}) - o(n_i)} + 1/2 \rfloor \quad (2.18)$$

where p is quantization precision. Here, we typically will be using the precision of $p = 5$ for this ratio, which is sufficient to preserve the perception of the rhythm [216]. Meek and Birmingham used similar approach with $p = 4$ precision [128], which gives slightly coarser representation.

2. **rhythm contour.** Like melodic intervals, IORs also can be reduced to contour information. However since rhythm does not have precise quantization, like melody with semitones, the general *ior* binning approach can be used with only three bins:

$$rc(n) = \begin{cases} -1 & \text{if } ior^r(n) < 1 \\ 0 & \text{if } ior^r(n) = 1 \\ +1 & \text{if } ior^r(n) > 1 \end{cases} \quad (2.19)$$

As Pardo and Birmingham suggested [144], rhythmic contour works surprisingly well in recognizing different melodies. This approach has been used, for example, by Uitdenbogerd and Yap [197]. Extended contours have also been proposed, but they are only another special cases of *ior*^r binning.

2.5 Analysis of Symbolic Music

Analysis of symbolic music usually focuses on creating algorithms that define similarity or distance between two music excerpts. Having a properly defined similarity measure allows for application of traditional data mining techniques for typical tasks, like classification, clustering and many others. Other approaches involve creating models, e.g., stochastic, generative or graph-based.

Typically, one can distinguish four main approaches to symbolic music analysis: string matching, geometric approaches, n-gram methods and hierarchical approaches. They all utilize the fact that events in music stream are precisely defined, where

each event, a note, is defined in usually two-dimensional space of pitch and time (duration), unlike for approaches dealing with audio data, where this information has to be inferred and it is assumed it might not be correct.

There has been a number of publications that summarize existing work. The following sections are based mainly, but not solely, on the analysis conducted by Rizo [163] and Orio [141].

2.5.1 String Matching

String matching techniques deal with note data as a continuous sequences of events, called strings, where the position of each note in the string, as well as the relationship with neighbouring notes are important. The matching process either finds similar areas or match the entire string returning a numerical score indicating how close two strings are together.

The easiest approach would be to use Hamming, Euclidean or Manhattan distance [119]. Hamming distance indicates the number of different symbols in two strings, Euclidean distance is the distance in an N -dimensional space, assuming \mathbf{a}_N and \mathbf{b}_N are vectors in this space, and Manhattan distance is the sum of differences between symbols at each position:

$$d_{hamming}(\mathbf{a}_N, \mathbf{b}_N) = |\{i | a_i \neq b_i, 1 \leq i \leq N\}| \quad (2.20)$$

$$d_{euclidean}(\mathbf{a}_N, \mathbf{b}_N) = \sqrt{\sum_{i=1}^N (a_i - b_i)^2} \quad (2.21)$$

$$d_{manhattan}(\mathbf{a}_N, \mathbf{b}_N) = \sum_{i=1}^N |a_i - b_i| \quad (2.22)$$

Although, computing the distance with any of these measures take only $O(n)$ time, they require strings with the same lengths, which is rarely the case, and any shift of notes in the input strings would cause a significant change of the distance. Despite that, those methods are sometimes used for music analysis [119, 120].

One of the issues with those approaches is, that even if the symbols in the input represent similar concepts, which are not exactly the same, then the outcome would

be similar to the situation where they were completely different. Unlike text letters and words, both pitch and rhythm dimensions are quantitative, which gives the ability to perform approximate matching. One of the approaches that matches on a symbol level called **δ - γ matching** [20, 21, 27], and it allows for small variations in values, further analyzed by Clifford and Iliopoulos [23].

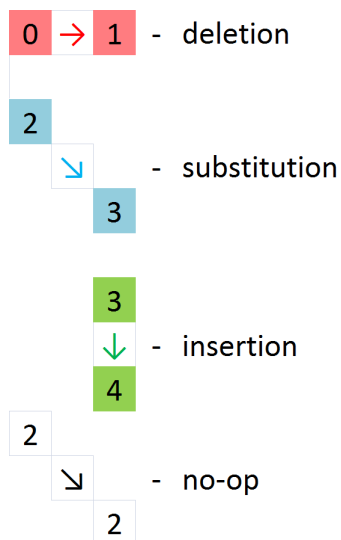
Typically, string matching techniques focus on computing edit distances between strings. Edit distance is the minimum cost to transform one string into another given a set of possible atomic, symbol-level operations, each with a certain predefined cost. The classical edit distance measure, called the Levenshtein distance, assumes three basic operations: *deletion*, *insertion* and *substitution*, all with the same cost of 1. An example how to transform two strings: ‘*knitten*’ and ‘*sitting*’ can be found on Figure 2.4(a). Levenshtein edit distance between these two strings is 4.

Edit distance can then be computed efficiently using dynamic programming with $O(N^2)$ time and can be seen as a shortest path problem through a grid graph defined by allowed edit operations (Figure 2.4(c)). Levenshtein edit distance allows for three transformative operations (Figure 2.4(b)). A non-transformative operation of rewriting the same symbol (*no-op*) is always allowed at no cost.

There are several approaches to using edit distances in music research. First of them is **global alignment**, using typically Needleman-Wunsch algorithm, used for aligning nucleotide sequences in bioinformatics, that computes the transformation cost from one string sequence to another. Levenshtein edit distance is just a variant of global alignment with insertion, substitution and deletion costs of 1. Since the entire sequence has to match, it is used only for matching of small excerpts and incipits, which are comparable in length [49, 145]. **Dynamic time warping** is another, frequently used variant of global alignment [80, 183], in principle not very different from global alignment, with one assumption, that the compared strings do not differ much and that the correct solution is found close to the diagonal of the dynamic programming grid graph, which allows for limiting the scope of the search and thus achieve sub-quadratic computational cost [80].

	k	n	i	t	t	e	n	
↓								
ε	n	i	t	t	e	n		
	↓							
	s	i	t	t	e	n		
					↓			
	s	i	t	t	i	n	ε	
							↓	
	s	i	t	t	i	n	g	

(a) string transformation



(b) available operations

		k	n	i	t	t	e	n
	0	1	2	3	4	5	6	7
s	1	1	2	3	4	5	6	7
i	2	2	2	2	3	4	5	6
t	3	3	3	3	2	3	4	5
t	4	4	4	4	3	2	3	4
i	5	5	5	4	4	3	3	4
n	6	6	5	5	5	4	4	3
g	7	7	6	6	6	5	5	4

(c) dynamic programming

Figure 2.4: How to compute Levenshtein edit distance between two strings, *'knitten'* and *'sitting'*: a) rewriting rules, b) available operations with their corresponding costs as differences between values in corresponding boxes. Deletion, substitution and insertion have a unit cost of 1, matching symbols (no-op) are accepted as zero cost. c) Shortest path indicating the cheapest transition from *'knitten'* to *'sitting'*.

Unlike global alignment, **local alignment** searches for similar continuous regions between two strings and does not aim to match the entire sequence. This can be achieved with the same dynamic programming framework, assigning different weights to each operation, i.e., awarding matching symbols (*no-op*) with positive score, penalizing insertion, deletion and substitution with negative score and not allowing the total score to go below 0. This approach can be used for searching for a phrase (e.g., user query, or a small excerpt) within a music piece. Uitdenbogerd and Zobel, in their comparison of different representation and analysis approaches [193], found the best results while using local alignment for matching. As with global alignment, utilizing parallels with bioinformatics allowed to port local-alignment-based techniques into music domain, e.g., MusicBLAST by Kilian and Hoos [95]. **Longest Common Subsequence (LCS)** is a simpler variant of local alignment, where only matching of a symbol is rewarded without penalizing of mismatches. This approach has been analyzed and used by Lemström et al [108] or Mäkinen et al. [120].

Until now, string matching techniques, apart from using specific music representation, do not differ from general text matching approaches. **Mongeau and Sankoff** were first to incorporate music knowledge into the design of a matching algorithm. They proposed a variable substitution cost based on consonance of intervals, and allow for more complex substitutions, i.e., fragmentation and consolidation, where several notes in one string correspond to a few in the other. To incorporate this in dynamic programming framework, one just has to allow for edges connecting nodes that do not correspond to neighbouring notes. Figure 2.5 illustrates how along with traditional edit operation (Figure 2.5 a, b, and c) one can add fragmentation of a tuplet (two notes) into triplet (three notes) (Figure 2.5 d), and consolidation of four notes to a single note (Figure 2.5 e).

Mongeau and Sankoff idea has been widely used for creating effective analyses of music content. Hu and Dannenberg [30, 80] and Grachten et al. [68] use the method directly while some modifications to the original idea could be found. Grachten et al. [69] evaluated optimal operation costs by using evolutionary optimization on

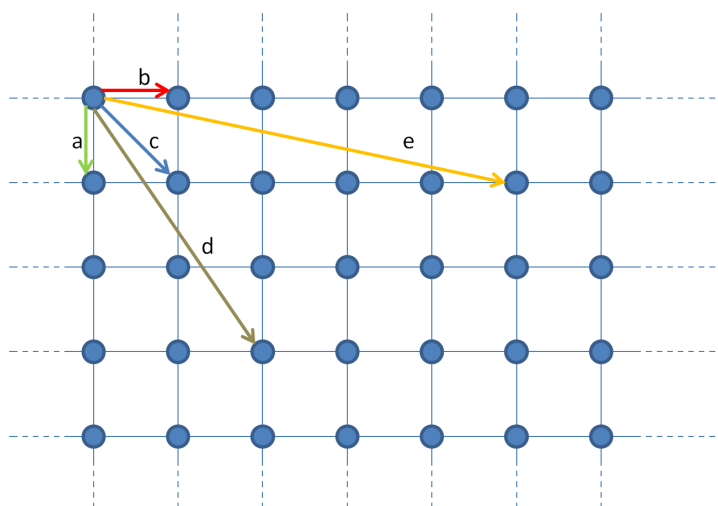


Figure 2.5: Edges on a grid graph corresponding to the following edit operations: a) insertion, b) deletion, c) substitution, d) fragmentation of a tuplet into triplet, e) consolidation of four notes into one.

melodies represented using Narmour’s implication/realization model [135]. Gomez et al. [61] implemented a simpler approach where substitution cost is not dependent on subjective consonance, but an objective distance, or pitch difference.

2.5.2 Geometric Approaches

The way concepts are represented, suggests sometimes the approach to tackle the data. Piano roll representation [121] of symbolic music may have triggered evolution of geometric approaches to symbolic music analysis, where notes are objects placed in pitch-onset-offset type of two dimensional space. Those methods typically handle polyphonic music naturally, which is often a problem for other methods.

Clausen et al. [22] proposed PROMS, a retrieval system where query is a sequence of points on a two-dimensional grid of MIDI pitch levels and onset times quantized to the closest sixteenth. The query is matched onto the target piece to check where the target melody overlaps with it. Wiggins et al. [207] proposed SIAMESE system that was able to find incomplete matches. Often, continuity of notes and melodies was reflected in representing notes as lines, indicating their duration [108, 109, 200]. Then, the similarity between two melodies was often the area between functions that

the melodies formed [115, 177] as initially suggested by Maidin [118]. Aloupis [1] extended this idea allowing for horizontal stretches of the query to accommodate for tempo and transposition invariance. However, minimizing the area between the curves can only be applied to monophonic data. Similar, time warping effect has been used by Laitinen and Lemström [101].

Earth Mover Distance is another technique proposed by Typke [185, 188, 190], where each note length is not represented as the length of a line, but as a weight (diameter) of a disc corresponding to this note, and the similarity between two melodies is the effort one has to put to transform (or move) points from one melody to the other. Since there is no notion of continuity, the model can be applied to polyphonic music without any change.

Detailed comparison between different geometric approaches with examples can be found in Lemström and Pienimäki [110].

2.5.3 N-gram Approaches

N-grams approaches are in assumptions very different from geometric, string matching, and other techniques used for symbolic music analysis. They are often inspired by research in Text Information Retrieval, which is based on the same principles and use similar methods. Primarily, information retrieval systems typically employ bag-of-words or bag-of-terms assumption, which states, that the order of terms in documents does not matter, so features (terms) are often combined in the form of sets. Music information retrieval and analysis systems can employ the same assumption, discarding information about the order of features (e.g., phrases, n-grams) in the analyzed music pieces. As a benefit, there is only one solution necessary to solve both problems of matching two sequences of similar length and searching for a query pattern with a large corpus of longer pieces. As a result of indexing process, retrieval of similar (or relevant) items is computationally cheaper than with other methods, because indexing escapes the need of scanning through the entire dataset to find a match for each query. With benefits, there are new issues that arise from applying

n-gram based approach. Text Information Retrieval typically uses words as features. With music, there is not clear distinction between lexical units, hence n-grams, which represent any arbitrary substrings of n symbols from the input sequence, are frequently used. The other issue is the discontinuity of the events, which result from the fact that the order of features does not matter. This can be partially addressed by using larger n to build n-grams that preserve more of the local context of each note event, however global context and long-range dependencies are usually lost with n-gram based techniques.

There are many approaches where an n-gram-based bag-of-terms approach and feature indexing are implicitly introduced, and often used as an intermediate representation to achieve other goals. The benefit of indexing features in music corpora to quickly retrieve data for further computation was used by Clausen et al. [22], where only single symbols, representing pitch related aspect for each note, were stored. Hoos et al. [79] presented an information retrieval system based on files in GUIDO music notation, where queries and documents were represented as non-deterministic finite state automata derived from sequences of pitches and durations using first-order Markov chains, which, in turn, is equivalent of building a probabilistic model based on bi-grams. Similar approach was proposed by Pardo et al. [145], where state transitions were determined by pairs of melodic intervals and IORs. Again, using a first-order Markov model implies a probabilistic model based on bi-grams. Li and Sleep [114] designed their melodic similarity measure based on calculating Kolmogorov complexity of note sequences using LZ78 algorithm, which resembles building a variable length n-gram profile while scanning through the note passage. Estimation of next symbol based on previous $n - 1$ (in this case n was fixed, $n = 4$) was also suggested and used by Serrano and Iñesta [167, 168] for pattern structure extraction.

What works best as a feature, or the music equivalent of word, was often an object of research and discussions. Melucci and Orio [129, 130] suggested segmenting input stream into non-overlapping phrases based on musicological knowledge. As suggested by Pienimäki [153], Neve and Orio [136] created segmentation approach

based on analysis of recurrent patterns, which resembles Li and Sleep's Kolmogorov complexity approach [114]. It turned out (Orio and Neve [142]), that all of those methods, including extraction of recurrent, musicologically oriented and perception-based patterns were outperform by a standard n-gram based extraction method using sliding, fixed-length window, which gives just a set of all possible substrings of length n . For detailed description on how a typical n-gram extraction process works, please refer to the Chapter 3.1.

There is a number of existing solutions that incorporate architecture and approaches typical for text processing tasks, like information retrieval, text data mining, or natural language processing. Typical bag-of-terms approach can be found in Uittenbogerd and Zobel [193, 198], where the performance of n-gram-based methods are shown to be on par with much more expensive alignment techniques. Downie [38, 41] analyzed n-gram based corpora and compared them to text corpora in terms of informetric features, showing resemblance between text words and music n-grams, but also noticing the differences. Downie and Nelson [40] proposed a retrieval system based on an of-the-shelf text retrieval solutions, where music n-grams were coded as text words and retrieved this way. Another system, that takes a general text retrieval solution, in this case Ponte and Croft's probabilistic retrieval engine based on language modeling [158], was proposed by Pickens [152]. He tested interval uni-grams and bi-grams for features achieving way better performance with bigrams, which shows the importance of combining basic feature components into bigger structures, like n-grams. Dannenberg and Hu [30] proposed a two-stage model where content is initially sieved with an n-gram based technique, which is then refined with Mongeau-Sankoff alignment. They pointed out outstanding high recall abilities of n-gram-based methods, with rather lower precision and ordering capabilities, thus additional alignment-driven postprocessing step. Doraisamy and R uger [33–37] proposed a retrieval system that handles any polyphonic symbolic music by collecting all possible monophonic n-grams

from any combination of notes within a certain window. This approach increases substantially the number of possible n-grams from each polyphonic excerpt, yet allowing for standard retrieval techniques to work with polyphonic data.

2.5.4 Hierarchical Approaches

Since music has its strong hierarchical structure on every level, there are approaches that try to create a parsing tree of music excerpts in the same way one parses sentences of natural languages with grammar rules, indicating dependencies between words. The main advantage, they have over string and n-gram methods, relies from the fact, that they consider not only dependencies based on the notes immediate neighbourhood. The reality is that notes build a similar structure to text, where dependencies can span the entire excerpt and often go beyond the excerpt level.

The initial proposal, Generative Theory of Tonal Music (GTTM) by Lerdahl and Jackendoff [111], was to create a grammar for music melodies and pieces in the same way they exist for natural languages, creating tree structures corresponding to parsing trees based on preference rules. It was based on an old idea of Schenkerian analysis, where a music piece could be observed on a number of levels where the top level consists of just a single object and each level is an elaboration of the previous level, down to the bottom, which consists of all notes from the given piece. The dependencies between those levels, drawn on a single figure, create a parsing tree of the piece. Since their model was in many places subjective, and based on common musicological sense, it have become only a very interesting theoretical system, although they indicated that it is possible to implement their reasoning. It has been finally done, after some simplifications, by Hirata and Matsuda [77] and Hamanaka et al. [70].

A similar theoretical approach, but simpler than GTTM, thus easier to implement, have been proposed by Narmour [135]. It is called Implication/Realization model and relies on the premise, that our perception of music depends on our expectations (implications) that build up while listening to the music and how they resolve (realization) — are they fulfilled (satisfaction) or not (surprise). The system was

based on a fixed number of basic units, that all melodies are build from. Identifying those patterns and segmenting based on them, has been used for measuring melodic similarity and retrieval by Grachten et al. [67–69].

Traditional linguistic approach to grammar learning from data using probabilities can be found in Bod [14, 15], where they used Essen Folksong Collection as input data. Explicit creation of Probabilistic Context Free Grammars (PCFGs) can be found in Gilbert and Conklin [46] where they built them based on Bach Chorales. A simpler approach have been proposed by Rizo and Iñesta [162], where trees are fit in the metric, i.e., depending on the meter, binary or ternary structure, and then they were used to calculate melodic similarity between monophonic and polyphonic melodies [161, 163].

2.5.5 Other Approaches

There are numerous approaches to symbolic music analysis that do not fall into any of the four categories introduced in the previous chapters, looking at the problem of symbolic music analysis from different perspectives.

Engelbrecht [47] treated melodies as progression functions seen as stochastic random variables. With this assumption, he was able to compute several typical statistical measures, like correlation, central moments, or mutual entropy, and he reported how one can use them to match melodic sequences.

Pinto [154, 155] viewed connection between notes as transitions in a graph of nodes representing chromatic encoding *base12* introduced earlier (see Equation 2.8). This may resemble 1-st order Markov chains analysis done by Hoos et al. [79] and Pardo et al. [145], which was similar to n-gram analysis, but here instead, Pinto uses typical graph-based methods for further analysis, comparing eigenvalues of Laplacians representing each melody graph.

There are some general approaches, that usually work for various kinds of data with little adjustments necessary to make them working for symbolic music. The main drawback of these one-size-fits-all methods is that the result is not self-explanatory. In

other words — we can plug the data, run the model and get very good results, but still it would not be easy to tell why this particular model works well, or why it does not. The valuable knowledge lies inside the model, a black box, and the results are hard to interpret. Some of those approaches use neural networks framework, and there have been research done, where it has been used with symbolic music data. Yahzong et al. [219] used neural networks to create an index of MIDI documents in the collection for further retrieval. Harford [72] used self-organizing maps to automatically segment and retrieve melodies based on both pitch and rhythm by creating separate models for each of two dimensions of melody. Neural networks are often used as evaluation method in conjunction with genetic algorithms for generation task [103, 139, 151]. In general, applying genetic algorithms to music data, mainly for generation task, has been widely used [12, 87, 92, 180] mostly with manual or supervised fitness evaluation. Since this problem will be approached in this dissertation, for detailed description of those approaches, please refer to section 4.1.1.

2.6 Linguistic Parallels in Music Research

There are discussions about the precise definition what a natural language is. One of the possible ways to define it could be found in Wikipedia, which states that it *“is any language which arises in an unpremeditated fashion as the result of the innate facility for language possessed by the human intellect”*. Not everybody agrees that music fits this definition, but music researchers, who know the rules of music, are usually more prone to agree with it. Orio [141], who refers to music as a language, points what kind of information music, as a language, conveys. Music, as well as text, has the symbolic representation that has its origins dated back in ancient times. Music and language are the only old human creative activities where symbolic representation is commonly used. Others, like painting, sculpture, dance did not have such common symbolic notation. Music notation cannot be directly ported into computers like text is, but this is only a representation issue that could be easily overcome. For instance, the argument that text can easily be split into words — the basic features for Natural

NLP level	Music research areas
phonetics	Waveform analysis, audio signals
phonology	Sound events identification
morphology	Score symbols, symbolization
syntax	N-grams, shallow reduction and parsing
semantics	Harmonics, phrase level, parsing
pragmatics	Phrases, voice leading
discourse	Interpretations, context of a piece

Table 2.2: NLP Levels with respective music research tasks.

Language Processing (NLP) and Information Retrieval (IR), which is not the case for music, can be countered if one mentions that there are natural languages that do not use anything to separate words, like Thai.

In order to treat music as a natural language, one has to show that music processing works on the same classes of problems as NLP does. One distinguishes certain levels of a text processing, listed in the Table 2.2. The research in NLP spans all those levels, from recording (a voice, speech) to understanding (the meaning of a text).

These levels also exist for music. Similarly to a natural language, music can be recorded and presented primarily as a waveform. On the ‘phonetics’ level one tries to investigate the structure of a sound, separate and distinguish between notes or instruments. However, music is much more complex in this area and sound recognition tasks are still facing basic problems.

The second very important similarity results from the fact both domains use symbolic notations. Music score also consists of characters which are called notes. Similarly to NLP’s morphology and syntax — music has hidden, grammar-like structure, and hidden rules — the harmony. It determines how to put words (notes) together, and how to build well-formed phrases with them. It also manages the musical meaning of a piece of which the basic exemplification is a progression of chords and notes. In the case of notes and their dependencies — we may talk about the syntax of the music, while in the case of chords or harmonic progressions — about the semantics of the certain phrase, or given the phrasing — the pragmatics of the excerpt. This is

very similar in its form to one of the main areas of NLP, which is grammatical analysis. The highest level of NLP (discourse) is also common in music in a form of ideas, desires or aspirations (romantic music) of a composer as well as images and actions behind it (program music). Dukas's "The Sorcerer's Apprentice" or Smetana's "Die Moldau" are very good examples of such music.

2.6.1 History of NLP and Music Research

The history of NLP reaches beginnings of the history of computation since it is believed that the ability of computers to process natural language as easily as we do will signal creation of intelligent machines. It comes from the assumption that the use of language is an inherent part of human cognitive abilities. Based on that, Alan Turing proposed in 1950 [184] a test, known as the Turing test, to determine if a machine is intelligent. Its main objective is that a truly intelligent machine could carry on a discussion with a human so that the latter could not recognize if he is talking to a human or to a machine. This definition of intelligence triggered the research in NLP with the ultimate goal to create such a conversational program. If it is a feasible goal, one has to actually and physically implement the way people think, reason and formulate thoughts.

At this point we can see a resemblance between human speech and music. Assuming music is a natural language, the Turing test would consist in generating musical pieces so that an expert could not recognize if an author is a machine or not. Why an expert and not a layman? Because it would be equivalent to a Turing test where an interlocutor does not know the language of the talk. We would call it a *Soft Turing Test* which, unlike for regular human natural languages, makes more sense for music.

Introduction of Grammars. There has been some pioneering work in both areas, NLP and Music Research. Some natural language analysis in a form of linguistics theories were made before the introduction of computing machines. As an early, pre-computer music research, one can point work to the of Heinrich Schenker with his Reduction theory in the beginnings of 20th century.

The approach to linguistic problems has changed substantially just after computers were invented. Interface to computers is textual and this is a natural format for computer files. Moreover, there was a very strong need for developing automatic machine translation. The beginning of this — the Noam Chomsky’s theory of context-free grammars for natural texts — dates back to 1956. Representing and processing music was not the top priority of that times. As a similar work in the field of music, one can point the book “A Generative Theory of Tonal Music” by Lerdahl and Jackendoff published in 1983. Both of this approaches deal with the respective areas in the same way — by introducing a formal grammar that may generate utterances in the given areas. However, there is substantial difference between the dates of publication of both works — nearly 30 years. The similar difference one can observe between the areas of music and textual Information Retrieval.

The period of ‘Look ma, no hands’. The early NLP researchers were very optimistic. The research was driven by the goal of developing automatic machine translation. Various systems were created but, although they worked perfectly on several, very limited examples, they were failing in the real-world applications. The research came to the point where nothing more could be achieved, and yet they haven’t created any robust system that will work on real data. This caused a crisis in the whole field. It looked that despite their complicated system, it is not possible to mimic human cognition in the area of natural languages.

It is likely the time where the music research has just come (Downie’s glass ceiling effect [42]). It is not that crucial as for natural languages, since one can still try to trick unexperienced listeners and thus pass the *Soft Turing Test* with the system that does not demonstrate the full understanding of the music matter it deals with. Another sign that the field of music research might be in this kind of situation is the introduction of seminars, where one asks about the future of the field (e.g., fMIR).

Present NLP. The previous knowledge-based approaches did not work well for many NLP tasks. Current research of NLP still deals with creating more precise

models that include more aspects and features of languages. However, research tends to shift toward stochastic NLP that uses various corpora-based methods, where the use of simple NLP techniques gives much better results than simple statistical analysis. NLP is also used to aid Information Retrieval systems, increasing their sensitivity to language-dependent features. Finally, great improvement has been done in machine translation, the first goal of NLP researchers, where Google Translate demonstrates the current state-of-the-art. This is the direction which current music research may follow, but this shift towards stochastic approaches in NLP would not be possible without emergence of large text corpora, which were available to researchers. It is still not the case, especially for symbolic music.

2.6.2 Music Linguistic Theories

Current research in music concentrates around Music Information Retrieval, both for the signal and symbolic music representations. In most cases it deals with basic issues how computers should deal with music data in general. The level of music interpretation does not go into semantics, probably because it is vague what the meaning in music is. However, one should notice that current text Information Retrieval benefits from using text semantics, by use of ontologies and relations between terms, dependencies between documents, or linguistic layer of text.

We would like to emphasize the work of Lerdahl and Jackendoff [111], who first described a generative approach that one can use toward the music. They describe it in a computational linguistics manner, using preference rules approach, mentioning that it could be possible to implement their rules in a working system. We had to wait for a long time for an implementation of their system, because they introduced several tough to define but fundamental concepts, that are easy to understand for humans, but hard to implement on a machine. A recent try, ATTA [70], deals with all the implementation issues by introducing several important limitations to the system, which makes the system not going beyond syntactic level by leaving behind harmony issues. Simpler model than the one proposed by Lerdahl and Jackendoff,

Implication/Realization model, was proposed by Narmour [135] and have been applied to several tasks by Grachten et al. [67–69].

Another generative approach based preference rules, which introduces very important component of modern NLP — probabilistic modelling, is described by Temperley [178]. Probabilities and corpus based statistics is an inherent part of all modern NLP theories hence probabilities can model the meaning of text by inferring dependencies within it [124]. This work reminds of the idea of probabilistic grammars introduced earlier for text [96] and proposed for music by Bod [15].

Statistical analysis is a very important component of NLP models and it has played (Cope [26]) and will play a major role in music research. In many cases, solutions to many problems that gave good results for texts, could give comparable results in music area. As an example, our n-gram method of authorship attribution developed for natural language texts [93] gave good results for composer recognition of musical pieces [215].

2.6.3 Future of Music Research

If the hypothesis that NLP and current computational music research operate on two similar fields is true, it could be beneficial for both domains. For instance, applications that span large number of levels of NLP (e.g., try to draw some high-level conclusions based on low-level music representations) could work better, if they focus only on fewer levels. As we have pointed out the layers of NLP, some of them are not that well covered for the music matter. Lots have been done in the areas of music ‘phonetics’, ‘phonology’ and ‘morphology’. We notice some recent work in the area of ‘semantics’ but there are no models in higher, much more interesting but complicated levels: ‘semantics’, ‘pragmatics’, and ‘discourse’. Those areas define the meaning of the data we deal with, the understanding of undergoing structure and the flow of composers ideas within a piece. In general, one can stack different applications given the structure of NLP, i.e., the output of a model that operates on syntactic level could be an input of a model operating on semantic level.

A few tasks that are relevant for music research and are well developed within NLP are sentiment analysis, genre classification, automatic summarization or idiom extraction. Other approach would be to enhance Music Information Retrieval with some semantic aspects of music matter — music ontologies with an application of shallow parsing (or alternatively, local reductions) to reach the level of current state-of-the-art of textual Information Retrieval. However, it is still not clear how to represent meaning of music in computational tasks, but in this case statistical approaches and data mining techniques may bring relevant tools to describe this phenomenon.

Chapter 3

Quantitative Analysis of Symbolic Music

This section provides introduction and analysis of content of a number of corpora, used further in this thesis for various tasks, as well as some other corpora, interesting for some unique reasons. Since music data is approached here in the same manner as text is for standard text information retrieval or natural language processing, it is important to point what text and music have in common and where they differ. The similarities would justify porting similar methods from one domain to another, and differences may indicate where one would need to look closely while designing a specific solution for music data. Downie indicates [41], that this kind of analysis, which he refers to as “informetric analysis”, allows not only to get an insight about the nature and quality of the data as well as the features that different corpora have in common, but also to estimate storage requirements, evaluate scalability and optimize system usage.

As a consequence of the analysis, the chapter ends with a series of experiments, where native text information retrieval techniques of measuring text similarity are applied directly to music corpora for composer classification task, as well as Music Information Retrieval Evaluation eXchange (MIREX) symbolic melodic similarity (SMS) challenge, which directly evaluates applicability of text-based methods of analysis to music data. Naturally, the differences between music and natural text exist, however, it is shown in this chapter how one can mitigate them in the preprocessing steps so the actual procedure of measuring similarity between two excerpts remains untouched.

3.1 N-gram-based Approach to Symbolic Music

If the hypothesis about the same provenience of music and text holds true, then music should have similar statistical properties as text as it arises from the same kind of cognitive process. In order to compare them, one needs a common feature space that removes representational differences between them. Text has a much more straightforward structure because it consists of a single flow of symbols while music can have (and usually has) multiple voices that sound at the same time with concurrencies possible even within each voice. This poses interpretation issues and performance bottlenecks as analyzing all possible passages leads to exponential explosion. This problem affects the entire domain of research in symbolic music as most of the methods have text analysis origin.

One of the methods to overcome these problems is to deal with one voice, and one note at a time. This is called the monophonic approach and it allows music to be treated as sequence data like text or genes. The opposite (polyphonic) approach, deals with all the notes from all voices at the same time, but this leads to computationally complex models. It is used in the tasks that require polyphonic data, like streaming [44], polyphonic pattern matching [24], or polyphonic music information retrieval [33, 41, 107, 193]. However, in most cases polyphonic music can be reduced to the monophonic level using various heuristics; the most frequently used one being the skyline method [193], which treats each voice separately and, given the voice, takes only the highest currently played note. Uitdenbogerd [194] analyzed a number of monophonization approaches concluding, that skyline method (dubbed all-mono) produces the most accurate results. This approach works best for precisely separated symbolically represented streams (where the only concurrency problem comprises chords and in-voice concurrencies), but it is also helpful in analyzing unprepared polyphonic data (where there is no voice separation).

The result of the linearization of music data is a series of symbols, each being a result of some function of these basic note features (e.g., pitch, duration, onset and offset time). In order to model and analyze these streams of symbols an n-gram

model is frequently used. Unlike for natural languages, where word boundaries are commonly seen (i.e., it is easy to separate basic lexicographical units (words) to be used for further analysis), music does not have these strict boundaries and this is where n-gram models become very useful.

N-grams are the manifestation of a Markov process where the probability of a symbol depends only on the probabilities of the preceding symbols. It is often further simplified to a Markov chain (of order n) where the probability of a symbol depends only on the previous $n - 1$ symbols. This can be represented as a ratio of two n-gram probabilities, or frequencies f :

$$P(x_i|x_{i-1}, \dots, x_1) = P(x_i|x_{i-1}, \dots, x_{i-n}) = \frac{P(X_i, \dots, X_{i-n})}{P(X_{i-1}, \dots, X_{i-n})} \simeq \frac{f(x_i, \dots, x_{i-n})}{f(x_{i-1}, \dots, x_{i-n})} \quad (3.1)$$

Typically, the general idea behind n-gram models is to count substrings of symbols of a certain length. However, what constitutes a good feature for n-gram analysis is the next problem to be solved. With the stream of notes, each having its own pitch and duration, there are numerous approaches proposed for how to convert it to a series of features to be later used as an input for such an analysis [17, 33, 41, 53, 157, 179, 193].

The straightforward approach to this problem is to take notes lengths and pitches and map them directly to features. This approach has a main caveat, as it does not satisfy tempo and transposition invariance requirement. Moreover, the same note with a certain pitch and duration may mean different things in different excerpts since its role also depends on the neighbouring notes.

This issue has been addressed in the previous work [34, 145, 215], which defines a relative representation, where each note transition is encoded by two numbers indicating relative pitch (melodic interval, as in 2.9) and relative change in duration (inter-onset interval ratio, IOR as per 2.18) with respect to the previous note. An n-gram then becomes a series of tuples indicating these relative changes.

More formally, for a given melody and a corresponding set of notes (i.e., MIDI events carrying information about note pitch and duration), one can create a set

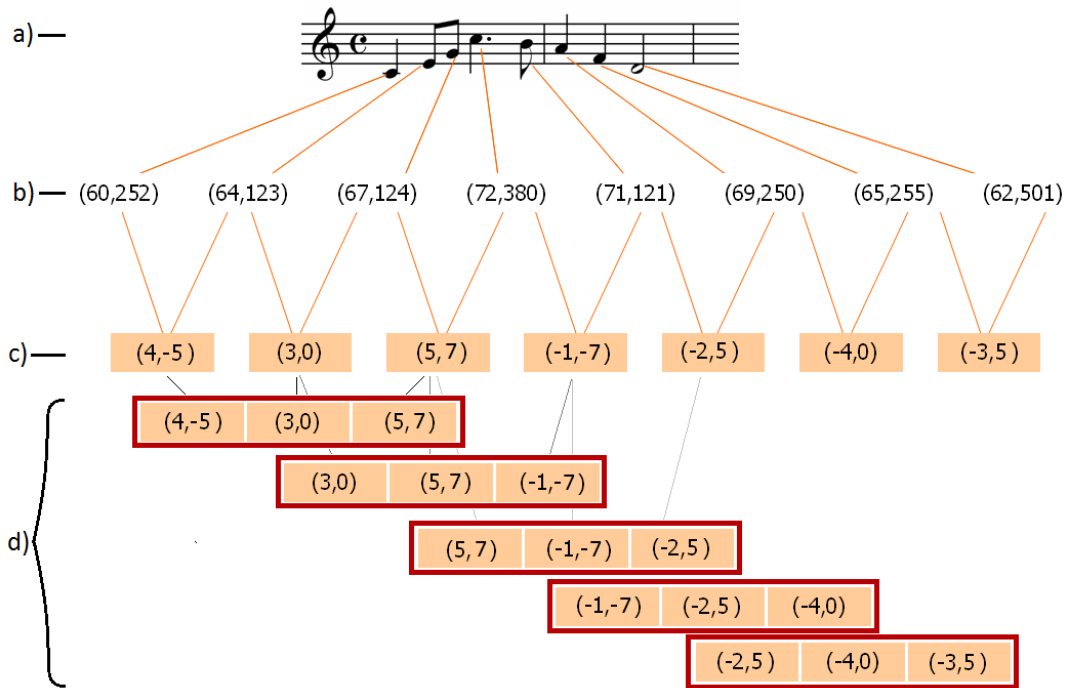


Figure 3.1: N-gram extraction process. a) Input monophonic melody. b) Raw MIDI features extracted from the input. Here: MIDI pitch and duration (in milliseconds). c) Uni-grams obtained by transforming each pair of basic features using Formula 3.2. d) Each N consecutive uni-grams form an n -gram, here of length 3. The initial melody is finally represented by the set of five 3-gram terms.

of features (dubbed uni-grams). Each uni-gram v_i represents the relative pitch and relative duration between two consecutive notes:

$$v_i = \langle mi(n_i), ior(n_i) \rangle \quad (3.2)$$

A simple n -gram's extraction process is shown in Figure 3.1. The same melody played in various tempos and in different pitches gives the same sequence of uni-grams, and therefore — n -grams. This is especially important in analyzing classical music, which often contains similar patterns that may be transposed to different keys and may be played with different paces.

One can notice that with this representation, for a melody of N notes one can obtain a sequence of uni-grams of length $N-1$. Each subsequence of n consecutive uni-grams can be then combined to form a single, n -gram term. N -gram sets representing

music documents or collections can be processed and analyzed in the same way as text with n-grams or words, or form a base for statistical models based on Markov assumption. It has been shown in our previous work [215], that this representation is useful for the composer classification task. Overall, this approach is widely adopted as fundamental, wherever bag-of-terms assumption is accepted.

3.2 Available Corpora

The common issue to deal with for most symbolic music researchers is that there is not many large, high-quality datasets available for analysis, as well as testing and comparing solutions to common problems. Some large audio and social metadata datasets are available, e.g., Million Song Dataset [11], since companies that deal with large amounts of music content, e.g., Last.fm or Youtube (Google), directly benefit from research on audio data. The benefits from symbolic music research are not directly transferable to new solutions and applications in their circumstances. Symbolic datasets require work of an expert musician to both sequence each individual document as well as annotate them with semantic information. They typically cannot be crowd-sourced as it is often done with text corpora, where many Web 2.0 services share their data for research purposes, like Wikipedia or Twitter. As a result, datasets used in symbolic music research are mostly small in size, created for a specific purpose, often through scavenging of online sources of MIDI data, e.g., 2006 and 2007 MIREX SMS mixed dataset of 15000 random web MIDI files [98], which often result in poor quality of data, or created as a collaborative effort of rather small music communities, e.g., String Quartetdataset, described in the following section.

High-quality, monophonic, or highly monophonised on voice level (i.e., after applying monophonisation procedure, like skyline method [194], the result sounds very similar to the original) files are the most desired for the purpose of the research presented in this dissertation. Most well-sequenced MIDI files, oriented not only for playback, that keep voices and instruments in separate channels, fit well this definition. However, it turns out, that they conform just a small fraction of all MIDI files

available in online resources like *classicalarchives.com*. So even, if one combines a dataset from the existing resources, it still does not mean, that the one would get a large, high-quality corpus with little effort. Several medium size music corpora will be analyzed in this chapter. They all contain accurately sequenced, monophonic or layered pieces of mostly classical music, either full compositions, or just themes or incipits.

The corpora of symbolic music will be analyzed along with a few text corpora, to find where symbolic music resembles natural languages and where it differs from them. Typically, research in information retrieval and natural language processing focuses on Western languages, mostly English, e.g., Downie compared statistical features of music n-gram corpora to English and “other Roman alphabet languages” [41]. In this chapter, another non-Western corpus will be analyzed along with an English corpus. This should allow looking at similarities and differences between natural languages and symbolic music from a broader perspective.

3.2.1 Symbolic Music Corpora

Piano Composers Dataset [`pcd`, `pcd_ex`]. Piano Composers Dataset contains 256 pieces of classical music, organized in 5 classes based on authorship. All the pieces in the corpus are either written directly for piano or for other keyboard instruments, like harpsichord or pipe organ, or they have been transcribed for a keyboard instrument, which makes the texture of the music data in the corpus uniform. The corpus have been put together for the previous work [216] to evaluate composer recognition approach. The items in the collection were selected manually by scavenging MIDI files from the Internet and then selecting those with voices separated into different channels with notes information likely reflecting the original music score. As an additional requirement, a piece qualified only if it sounded similarly to the original even after applying skyline linerization process [193] indicating, that monophonization does not affect the overall music perception of the composition. The main classes of `pcd` dataset contain the following pieces:

- **bach** — 113 pieces of Jan Sebastian Bach, consisting of 24 Preludes and Fugues from “das Wohlthempierierte Klavier” vol. 1 (BWV 846–869) (48 pieces), 3 Preludes and Fugues from vol. 2 (BWV 871, 872, 888) (4 pieces), 34 parts of various Cantatas, 20 pieces from 15 Inventions set (15 arranged trios by Straube and Reger, 5 selected originals) (BWV 772–786), 5 pieces from Anna Magdalena Notebook, second part of Italian Concerto BWV 971, and Duetto No. 2 (BWV 803).
- **beethoven** — 45 pieces of Ludwig van Beethoven, mostly Piano Sonatas (30 pieces), but also Variations (2 pieces), 6 Bagatellen op. 33, Rondos (3 pieces), Romances (2 pieces), Sonatina op. 79, and Bagatelle Für Elise.
- **chopin** — 58 pieces of Frederic Chopin including: Preludes op. 28 (24 pieces), Etudes (8 pieces), Nocturnes (6 pieces), Mazurkas (8 pieces), Waltzes (2 pieces), Fantasie-Impromptu op. 66 (2 renditions), Ballade op. 23, Scherzo op. 31, Piano Sonata op 35 (4 pieces), Polonaise op. 53, and Concerto op. 11, part 2.
- **mozart** — 17 pieces of Wolfgang Amadeus Mozart including: Sonatas (9 pieces), Rondos (2 pieces), Piano Miniatures (4 pieces), Variations KV 265, and Overture to KV 384 (arranged for piano).
- **schubert** — 23 pieces of Franz Schubert including: Sonatas (16 pieces), Moments Musicaux op. 94 (5 pieces), Klavierstücke D. 946 No. 1, and Ave Maria D.839.

There is an extended version of this dataset, `pcd_ex`, with a number additional composers, with fewer pieces in the collection, which gives in total 366 pieces:

- 39 pieces of Johann Burgmüller: Etudes Faciles op. 100 (22 pieces), Etudes op. 105 (9 pieces), and Etudes op. 109 (8 pieces),
- 11 pieces of Muzio Clementi: Sonatinas (8 pieces), and Etudes (3 pieces),
- 5 various pieces of Georg Händel,

- 16 pieces of Modest Musorgsky (Pictures at an Exhibition),
- 12 pieces of Robert Schumann: Scenes from Childhood op. 15 (5 pieces), Album for the Young op. 68 (4 pieces), Moment Musicaux op. 94 (2 pieces), and Fantasie op. 12 No. 3,
- 11 pieces of Alexander Scriabin: Preludes (8 pieces), Etude op. 8 No. 5, Scherzo op. 46, and Nocturne op. 9 No. 2, and
- 16 pieces of other composers: Luigi Boccherini, Alexander Borodin, Claude Debussy (2 pieces), Léo Delibes, Edvard Grieg, Joseph Haydn (2 pieces), Scott Joplin (3 pieces), Ferenc Liszt, Felix Mendelssohn (2 pieces), Johann Pachelbel, and Richard Winkelmann.

Haydn/Mozart String Quartets [`string`, `string_rh`]. String Quartets dataset contains all string quartets of Joseph Haydn and Wolfgang Amadeus Mozart. They both lived in the same times (Haydn was born in 1732 and died in 1809, Mozart was born in 1756 and died in 1791) and in the same place — Vienna, which made their style very similar. Since the corpus contains only one music form, this makes it very uniform. The data come from Center for Computer Assisted Research in the Humanities (CCARH) at Stanford University, where they make them available in `kern` format. The files were then converted to MIDI using Humdrum toolkit, ensuring separation of voices into different tracks, to allow for direct use of monophonic methods using string or n-gram approaches. The dataset consists of 210 Haydn pieces and 82 from Mozart. Imbalance of the size of datasets reflects the difference in life time of both composers. Although it is not typically a strict rule, it holds true in this particular case.

Hillewaere et al. [75] suggested modifications to the original `string` datasets, by removing late Haydn quartets written after Mozart death, indicating that they can be too easy to classify. They also added a few pieces of the same type to Mozart data, resulting in more balanced dataset. Since they approached automatic classification task, based on the modified dataset, it would be interesting to observe, how the

performance of classifiers changes between those datasets. Their dataset has been incorporated for classification analysis as an additional `string_rh` dataset.

J.S. Bach’s Well Tempered Klavier, vol 1 [`wtk`, `wtk_rand`]. `wtk` dataset contains 48 preludes and fugues from volume 1 of “Das Wohltemperierte Klavier” by Johann Sebastian Bach. Although these pieces are contained within `pcd` dataset, this is a different rendition, where the quality of the data was checked to ensure that each voice has its separate track. Fugues typically have 2 to 5 strictly monophonic voices (dubbed voci), although being scripted on a 2 staff system for keyboard instruments. In polyphonic (music theory), baroque music, and especially in fugues, each voice is equally important and typically does not contain any accompaniment or is in subordinate relation to any other voice, so `wtk` dataset have been selected as a training corpus as a good sample of melody leading for melody generation task, introduced in the next chapter.

`wtk_rand` dataset is an artificial dataset, based on `wtk` dataset. It was created from notes (melodic intervals and IORs) statistics obtained from `wtk` corpus, by taking the probability of each pair of interval and IOR, as well as the probabilities of end-of-track and end-of-file markers, and then drawing random symbols from this distribution, one at a time, and appending them to the output. Analysing such dataset would allow to observe, which statistic features of music corpora come from the basic frequencies of symbols in the alphabet, and which have deeper origin. 48 pieces were drawn with this method, the same number as in the original `wtk` dataset. It turned out, rather unsurprisingly, that even without direct control over the size of each file, we have obtained a dataset with a very similar size as the base one.

Répertoire International des Sources Musicales, MIREX subset [`rism`]. Répertoire International des Sources Musicales is a non-profit organization, founded in 1952, which aim is to document music sources all over the world from all times. Their Series A/II catalogue contains information about handwritten manuscripts with

more than 700,000 entries, each containing an incipit (the excerpt starting at the beginning of the piece) reflecting the content of each manuscript. Currently, the access to the database is provided by RISM through their queryable catalogue, although as for now, one can query it solely in text mode and no melodic search is available.

A subset of RISM A/II dataset, provided by UK RISM division, have been used in Music Information Retrieval Evaluation eXchange (MIREX) challenge for Symbolic Melodic Similarity (SMS) in 2005 and 2006. It consists of 2 datasets, training and evaluation, containing in total 1110 purely monophonic incipits in MIDI format. The subsets were supplied by Typke et al. [186] with 22 queries (11 for training and 11 for evaluation) along with relevance judgements of similarity between each query and retrieved items from the collection, obtained via human evaluation of 2005 and 2006 MIREX SMS challenge. Due to the availability of query relevance data, this dataset is often used as an internal benchmark by participants of SMS MIREX challenge. In this dissertation, it is used to investigate which aspects are important while designing a melodic similarity measure using n-gram approaches.

Essen Folksongs Collection [essen]. Essen Folksongs Collection (EFC) is a result of joint initiative of German and Polish ethnomusicologists to preserve musical heritage of folk songs from nations around the world. Currently, the holdings reached 20,000 pieces mainly from Germany, China and Poland. Each item in the collection is a monophonic melody encoded in EsAC format, with conversions to other notations, like kern and MIDI available. Since the collection does not have the licensing issues, that came with RISM dataset, it has been adopted as an evaluation dataset for MIREX SMS task in 2007, 2010, 2011 and 2012. The dataset used for MIREX task is a subset containing 5274 melodies in MIDI format.

Beatles Dataset [beatles]. Although this dissertation is focused on classical music, it is usually desirable to verify how well algorithms designed with one music genre in mind act on a different type of music. Additionally, the analysis of music content based solely on classical music would give non-representative results, as modern pop

music becomes also a very important part of our music heritage. For those reasons, it seemed important to include in the analysis other, non-classical corpus, which in this case is a collection of 209 Beatles songs. Apart from the purpose of comparing different types of music in this chapter, items from this dataset are used to observe how visualization and segmentation algorithms introduced in this thesis and designed with classical music in mind, perform on a different type of music.

3.2.2 Textual Corpora

Analysis of text corpora is not an essential part of research on music but it seems reasonable to use them in comparison with music datasets to highlight similarities and identify differences between natural languages and music. Typically, music is compared against Western languages, like English [41]. However, to broaden the scope, we also included a corpus of Thai language, which shares some interesting features with music.

Wikipedia [`wiki` and `wiki_rand`]. Wikipedia dataset `wiki` contains the first (by ID) 10,000 articles from English Wikipedia acquired from 2012/03/12 snapshot downloaded from Wikipedia backup site (`download.wikimedia.org`). The dataset have been processed and cleaned using `wikiprep` program [57]. As a result, it yielded about 90.1MB of clean (markup removed) text.

To create `wiki_rand` corpus, the same method, which was used to create `wtk_rand`, has been applied. The probabilities of each letter, space, punctuation symbol, and end-of-document marker have been estimated using `wiki` dataset and another 10,000 documents have been drawn, one symbol at a time, based on the underlying probability distribution of symbols. Here is a small sample of text created as a result of such process:

```
thonarae e inn .inlc eceatts p awcht2glni tp  l,f t6rn s-lrae:e
tryms2 ,eaopc eol)(  h de tleuSeisowiIeaMuDi'nprA"eIdhen
r2Ftr v Onu2 hdl,he%9l3sC relomt Selcacbmthdhaeo r
```

irtionibCJrun hr0m*cl2fnau.

The interesting point would be to observe, which measurable features of written text result from symbols distribution and which have deeper reason, and among them — which are shared with music, indicating whether symbolic music might be perceived, from the analysis perspective, just as a specific natural language.

Thai ORCHID Corpus [orchid]. ORCHID project (Open Linguistic Resources CHannelled toward InterDisciplinary research) was initiated in 1996 to create a Thai part-of-speech tagged corpus, mainly to support research in natural language processing. The corpus contains 164 documents, mainly scientific documents from the field of computer science. For the purpose of comparing various text and music corpora, all the tagging information have been removed, leaving just the pure, Thai text.

Thai language shares a number of properties with symbolic music, which are different from properties of many other languages. Thai alphabet consist of a small number of phonograms (letters), each representing a phoneme or a speech sound (like in English), but different from languages like Chinese, which uses a large number of logograms. However, like in Chinese, word boundaries are not indicated in the text, which poses a problem similar to phrase detection in music. The second similarity with music notation, results from the fact, that any of the 70 base characters can be altered with one or more of 17 combining characters (as defined by TIS-620 standard) which gives a large number of combinations representing letters, whereas languages based on Latin alphabet are typically limited to a fixed, small number of possible characters.

3.3 PERL MIDI::Corpus Module

In order to effectively process and analyze large amounts of data for various tasks, one needs to obtain a set of tools to handle the data, which are shared between those tasks. PERL programming language has a number of built-in tools, that allow easy access, handling and analyzing of text data, which is already in a quite simple format.

MIDI is a much more complicated, binary format, and symbolic music itself is not tailored to fit basic data structures and algorithms, like text.

`MIDI::Corpus` addressed those issues by providing object-oriented interface to handling and analyzing corpora of symbolic music in MIDI files. It does not handle MIDI files internally, but instead, uses `MIDI-Perl` suite. It provides a set of high-level functions to load corpora of MIDI files, manipulate them to obtain various statistics, and exports a number of functions for measuring similarity between corpora.

3.3.1 Synopsis

Here is a sample code using `MIDI::Corpus` module to retrieve 10 most similar pieces to the query:

```
#!/usr/bin/perl
use MIDI::Corpus qw/cosine/;
use strict;

opendir DATA, "beatles";
my $corpus=MIDI::Corpus::new;
$corpus->add("beatles/$_ [as] $_")
    for grep {/\.midi?$/i} readdir DATA;
closedir DATA;
$corpus->add("unknown_song.mid [as] query");
die $corpus->error unless $corpus->ok;

my %sim;
$sim{$_}=cosine(4, 'query', $_, $corpus) foreach $corpus->items;
delete $sim{'query'};

my @ranked = sort {$sim{$b}<=>$sim{$a}} keys %sim;

print "$_: $sim{$_}\n" for @ranked[0..9];
```

3.3.2 Corpus Loading

Module provides a `MIDI::Corpus` object that encapsulates main module information and allows for access to module methods. Here, this object is typically referred to

through a `$corpus` variable. The following functions can be used to create an object, set its parameters and fill it with data.

`new(@files)` — Creates a new object and defines music features as combined melodic intervals and IORs. If any arguments are supplied, it treats them as a list of files to be added to the collection. Returns itself for stacking if files were added successfully or `undef`, if an error occurred while adding files.

```
my @files = qw/ foo.mid bar.mid baz.mid /;
my $corpus = MIDI::Corpus::new(@files);
die "error while loading corpus" unless $corpus;
```

`setf(&function)` — Changes how features are extracted from the raw flow of notes and unigrams. The supplied function should take two list references, to **notes** and **unigrams**. Each element in **notes** list is a 3-tuple indicating absolute parameters: start time, MIDI pitch, and duration. Each element in **unigrams** list is a pair of relative parameters: melodic interval and IOR. The function should return scalar value uniquely identifying n-gram from those input notes. The call to `setf` should immediately follow `new`, prior to adding any elements to the corpus.

```
sub melodic_interval_ngrams
{
  my ($notes, $ugrams) = @_;
  my @intervals;
  foreach my $ugram (@$ugrams)
  {
    my ($mi,$ior) = @$ugram;
    push @intervals, $mi;
  }
  return join ';', @intervals;
}
my $corpus = MIDI::Corpus::new;
$corpus->setf(&melodic_interval_ngrams);
    #setting melodic interval n-grams

#or just

my $corpus = MIDI::Corpus::new()->setf(sub{
```

```

        join ', ', map {$$_[0]} @{$_[1]};
    });
#setting IOR n-grams

```

`add(@files)` — Adds items to the collection. Accepts a list of document descriptors. They contain either paths to MIDI files or paths with optional labels, under which they should be identified, delimited with `[as]` keyword. If no labels are given, files will be identified by their full filename and path. Returns `undef` if there was a problem with any of the files, or itself upon success.

```

my $corpus = MIDI::Corpus::new;

my @files = qw/foo.mid bar.mid baz.mid/;
$corpus->add(@files);

open DIR, "very/long/path";
my @midi = readdir DIR;
close DIR;

$corpus->add(@midi); #files will have long labels

$corpus->add("very/long/path/$_ [as] $_") for @midi;
#shorter labels containing just file names

```

`in_collection($label)` — Checks if a file under label supplied as an argument exists in the collection.

```

opendir DATA, "beatles";
my $corpus=MIDI::Corpus::new;
$corpus->add("beatles/$_ [as] $_")
    for grep {/\.midi?$/i} readdir DATA;
closedir DATA;
if ($corpus->in_collection("Yesterday.mid"))
{
    print "'Yesterday' is in the collection";
}
else
{
    print "'Yesterday' is missing";
}

```

items — in list context, returns a list of labels of items in the collection. In scalar context, returns number of items in the collection. In void context, returns `undef`. See Synopsis 3.3.1 for usage.

3.3.3 Corpus Status

ok — returns true if there was no issues with the corpus, and false otherwise.

error — returns an error message if there was a problem with the corpus. When called, resets error status, so subsequent calls to **ok** return **true** until a new problem is encountered. Used in conjunction with **ok**.

explicate — prints basic information about items in the collection.

```
my $corpus=MIDI::Corpus::new;
#existing MIDI files: ./foo.mid ./bar.mid (corrupted)
sub check
{
    $corpus->add(@_);
    print ($corpus->ok)?'OK':$corpus->error;
}
check 'foo.mid';           # OK
check 'foo.mid';           # foo.mid already in the collection
check 'foo.mid [as] other'; # OK
check 'baz.mid';           # Cannot locate baz.mid midi file
check 'bar.mid';           # Cannot load bar.mid: <error description>
$corpus->explicate;
#---
#Corpus has following features:
#  number of files: 2
#    0 file (foo.mid) has 3 tracks
#    1 file (other) has 3 tracks
#  number of statistics: 0
#---
```

3.3.4 Playback

N-gram and track playback options are available only on Windows systems, since they require Win32::MIDI module to be installed, and loaded.

`playable` — enables playback functions

`play_track($label,$track)` — plays content of file of label indicated by the first argument for track number indicated by the second argument.

`play_ngram(\@unigrams)` — plays the content of an arbitrary sequence of unigrams containing melodic interval and IOR information

```
use MIDI::Corpus;
my $corpus=MIDI::Corpus::new('foo.mid')->playable;

$corpus->play_track('foo.mid',0);

#twinkle, twinkle, little star...
$corpus->play_ngram([
    [0,0],[7,0],[0,0],[2,0],[0,0],[-2,5],
    [-2,-5],[0,0],[-1,0],[0,0],[-2,0],[0,0],[-2,5]
]);
```

3.3.5 Data Functions

`raw($label, ?$type?)` — This function allows to retrieve files content if one wants to analyze symbolic music data in a different way, than provided by analysis functions of `MIDI::Corpus` package, yet still benefit from easy MIDI files handling. Note, that results are already linearized using skyline method, so no polyphonic processing can be done with `MIDI::Corpus` package. The function takes one mandatory argument, label of the file. If second argument is provided, and it is `'ugrams'`, it returns array of unigrams from this label. If the second argument is `'notes'`, a list of absolute notes information is returned. If second argument is not provided, a reference to the entire data structure, corresponding to this item in the collection, is returned. The following example shows, how one can obtain interval statistic using `raw` function.

```
use strict;
use MIDI::Corpus;
use List::Util qw/min max/;

my @files = qw/ foo.mid bar.mid baz.mid /;
```

```

my $corpus = MIDI::Corpus::new(@files);
die $corpus->error unless $corpus->ok;

my %stat;
foreach my $item ($corpus->items)
{
    foreach my $note ($corpus->raw($item,'ugrams'))
    {
        $stat{$$note[0]}++;
    }
}
print "$_ exists ",0+$stat{$_}," times\n"
    for (min(keys %stat)..max(keys %stat));

```

3.3.6 Analysis Functions

`create(\%options)` — creates n-gram statistics for items in the collection and returns a reference to a data structure with statistics, typically a hash reference. If an error occurred, `undef` is returned and `error` is set. The statistics is also saved within a corpus for further use, so to free the memory, one needs to `undef`'ine the corpus after statistics has been used. Method accepts options in key-value fashion. Mandatory options are `'length'`, which is any positive integer value indicating n-gram length, and `'type'` — indicating type of statistics required for the task. Available options for `'type'` parameter are:

- `count` — calculates how many times each n-gram occurs. Returns simple hash reference, with n-grams as keys, and counts as values.
- `probability` — calculates probability of each n-gram in the corpus. Returns a hash reference.
- `lobability` — calculates logarithms of probability of each n-gram. Returns a hash reference.
- `probexists` — calculates probability of terms assuming binary weights (1 — exists, 0 — otherwise). As a result, all terms will have the same weights, normalized to the total number of different terms. Returns a hash reference.

- `tf` — calculates term counts for each document separately. Returns two-tier hash, with each count associated with document label and n-gram.
- `df` — calculates in how many documents each n-gram occurs. Returns a hash reference.
- `tfidf` — calculates `if.idf` score for each n-gram, according to Equation 3.19. Returns structure similar to `tf`.
- `bm25` — calculates `bm25` score for each n-gram, according to Equation 3.20. Returns structure similar to `tf`.

Other parameters include:

- `'function'` — temporarily overrides default combining function. Takes the same function type as `setf` method. The global combining function remains unchanged.
- `'name'` — optional identifier for created statistics. By default, statistics are identified by `type` and `length`. If you want to create a different statistics with the same `type` and `length`, but different other optional parameters, you should specify its name.
- `'force'` — if set, calculations will be redone with new parameters, even if the statistics of the same identification exists. The old statistics is deleted and the memory it took — freed. This allows to perform series of analysis on the same corpora, without taking more memory, each time a new analysis is performed.
- `'k'` and `'b'` — set custom k and b parameters for `bm25` weighting function overriding default $k = 2$ and $b = 0.75$. See Equation 3.20 for details.

```
use MIDI::Corpus;
$corpus = MIDI::Corpus::new(qw/ foo.mid bar.mid baz.mid /);
#counting bi-grams of combined MI and IOR across the corpus
$count = $corpus->create('type'=>'count', 'length'=>2);
#counting melodic interval bi-grams
```

```

$counts = $corpus->create(qw/ type count length 2 /,
    'function'=>sub{return join ', ', map {$_[0]} @{$_[1]}});
    #wrong - 'count 2' already exists
$counts = $corpus->create(qw/ type count length 2 force 1/,
    'function'=>sub{return join ', ', map {$_[0]} @{$_[1]}});
    #correct - 'count 2' overwritten and recalculated
$counts = $corpus->create(qw/ type count length 2 name mi_stat /,
    'function'=>sub{return join ', ', map {$_[0]} @{$_[1]}});
    #correct - new entry created, keeping previous statistics
print "most common melodic bi-gram occurred ",
    max(values %$counts), " times\n";

```

`lz78($label, ?\ %dictionary?, %options)` — Estimates Kolmogorov complexity using lz78 algorithm of document specified by the first argument. If dictionary is also specified, calculates conditional complexity, using supplied dictionary. Returns the difference of dictionary sizes before and after application of lz78 algorithm. Changes dictionary by including new dictionary entries, so calls to `lz78` can be stacked to build a single dictionary for multiple documents. It takes optional parameter, `function` and `length`, in the same manner as `create` function, although `length` carries minimal sense for complexity estimation. For details on the method, see Section 3.4.5.

```

use MIDI::Corpus;

$foo = MIDI::Corpus::new('foo1.mid', 'foo2.mid');
$foo->lz78($_, \%foo_dict) for $foo->items;

$bar = MIDI::Corpus::new('bar1.mid', 'bar2.mid');
$bar->lz78($_, \%bar_dict) for $bar->items;

$baz = MIDI::Corpus::new('baz.mid [as] qux');

$foo_score = $baz->lz78('qux', \%foo_dict);
$bar_score = $baz->lz78('qux', \%bar_dict);

say "baz looks more like ", ($foo_score < $bar_score)? 'foo': 'bar';

```

3.3.7 Similarity Measures

MIDI::Corpus module exports a number of functions allowing for similarity calculations of items within a corpus or across different corpora. To use them, one has to import them in one's own script next to `use` command. All functions share common calling style:

```
function($length,$doc1,$doc2,$corpus1,?$corpus2?)
```

where *\$length* specifies n-gram length, *\$doc1* and *\$doc2* are labels of two documents. If they come from the same corpus, only *\$corpus1* is specified, otherwise *\$corpus2* should be specified as well. They use built-in corpora statistics, so the first call to any similarity measure may take longer to run, required to compute relevant corpus statistics, like `tf.idf` or `count`, depending on the measure. See Synopsis 3.3.1 for usage.

The following measures are implemented:

- **binary** — returns the number of terms two documents share in common. See Equations 3.11 and 3.16 for details.
- **cosine** — computes cosine similarity using standard term weighting method based on n-gram counts. See Equations 3.13 and 3.17 for details.
- **binary_cosine** — computes cosine similarity with binary term weighting, discarding how many times an n-gram occurs in a document.
- **euclidean** — computes euclidean distance between two documents.
- **tfidf** — computes dot product between two documents with *tf.idf* term weighting. See Equation 3.19 for details.
- **tfidf_norm** — computes cosine similarity between two documents with *tf.idf* term weighting. Essentially, it is normalized version of `tf.idf` method.
- **bm25** — computes *bm25* score between two documents in a similar fashion as `tfidf`, but with *bm25* term weighting method. Consult Equation 3.20 for details.

- `bm25_norm` — normalized version of `bm25`.
- `cng` — computes CNG similarity score, as defined in Equations 3.14 and 3.18.
- `cng_profile` — computes CNG similarity score between entire profiles of two corpora, like in `cng` function. Labels of `$doc1` and `$doc2` are ignored.

3.4 Statistical Analysis of Music Corpora

3.4.1 Size Statistics

The initial analysis focuses on the sizes of the corpora, as well as the sizes of documents and distribution of terms in each corpus. Typically for bag-of-terms approaches, it is important, how many distinct terms are in each document and in a collection as a whole. This allows to estimate the size of the search index, so this information has been included in our analysis as well.

Table 3.1 contains size information for all the datasets introduced in the previous section, along with analyses on text datasets on letter and word level. Most datasets sizes are in the same range, from 246k symbols (`essen`) to 993k (`orchid`) with a few small ones (`wtk` and `rism`) and much bigger Wikipedia-based datasets. The `orchid` corpus has a much bigger variety of symbols, comparing to English text in `wiki`, yet still not as complex as music datasets. We can also observe, that randomization procedure, applied to `wtk` and `wiki` datasets, does not change much the volume of the data and statistics on symbol level, but it does change greatly the number of words in `wiki_randw` dataset, which should also be the case for music n-grams, with $n > 1$.

3.4.2 Distribution of Intervals and IORs

Another analysis typically performed on symbolic music corpora, involves finding the distribution of symbol probabilities. Since both melodic intervals and inter-onset interval ratios (IORs) are represented with numbers, it is possible to plot them on a numerical scale. This can not be done with text letters or words. However, since

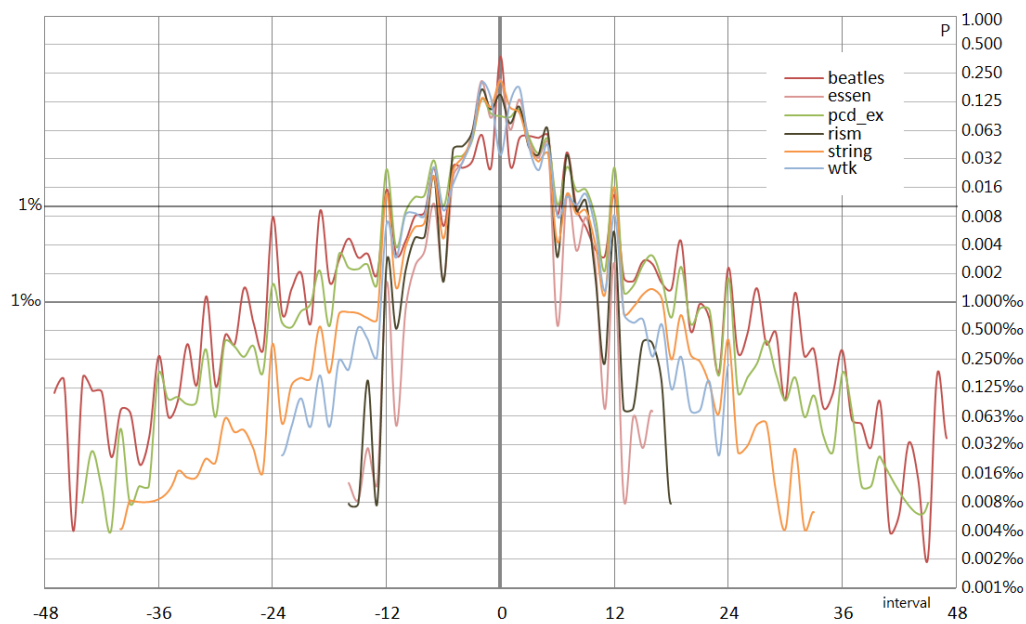
Corpus	$ D $	$ T $	$ \{t_i\} $	$ \bar{d}_i $	$P_{10}^{d_i}-P_{50}^{d_i}-P_{90}^{d_i}$	$ \bar{t}_{d_i} $	$P_{10}^{t_{d_i}}-P_{50}^{t_{d_i}}-P_{90}^{t_{d_i}}$
beatles	209	529k	4416	2530	1208-2273-4443	213	108-173-364
essen	5274	246k	494	46.6	27-43-72	20.5	13-20-28
pcd	256	421k	3299	1644	300-1025-4434	137	46-110-239
pcd_ex	366	525k	3440	1434	279-885-3561	122	46-95-221
rism	1110	13.7k	352	12.4	7-12-19	9.2	6-9-13
string	292	511k	2065	1751	583-1618-3099	178	109-174-256
wtk	48	42.7k	933	890	514-791-1383	110	60-112-173
wtk_rand	48	39.6k	747	826	42-624-1774	123	25-126-226
orchid	164	993k	551	6056	663-5877-11657	181	91-195-232
wiki	10k	91.0M	38	9098	985-3868-23722	35.3	31-37-37
wiki_rand	10k	93.3M	38	9332	929-6444-21790	36.6	35-37-38
wiki _w	10k	15.6M	310k	1556	164-692-3951	544	103-323-1290
wiki_rand _w	10k	15.7M	6.38M	1569	157-1081-3663	1167	140-848-2646

Table 3.1: Size statistics of analyzed corpora. $|D|$ — number of documents in the collection, $|T|$ — total number of terms: unigrams (for music databases), or letters and words (for text databases), in the collection, $|\{t_i\}|$ — total number of distinct term (i.e., size of the dictionary), $|\bar{d}_i|$ — average length (number of terms) of a document, $P_{10}^{d_i}-P_{50}^{d_i}-P_{90}^{d_i}$ — distribution of document lengths: 10th percentile, 50th percentile (median), and 90th percentile, $|\bar{t}_{d_i}|$ — average number of different terms in a document, $P_{10}^{t_{d_i}}-P_{50}^{t_{d_i}}-P_{90}^{t_{d_i}}$ — distribution of the number of different terms per document. Text databases analysis is available for characters (**wiki** and **wiki_rand**), and words (**wiki_w** and **wiki_rand_w**).

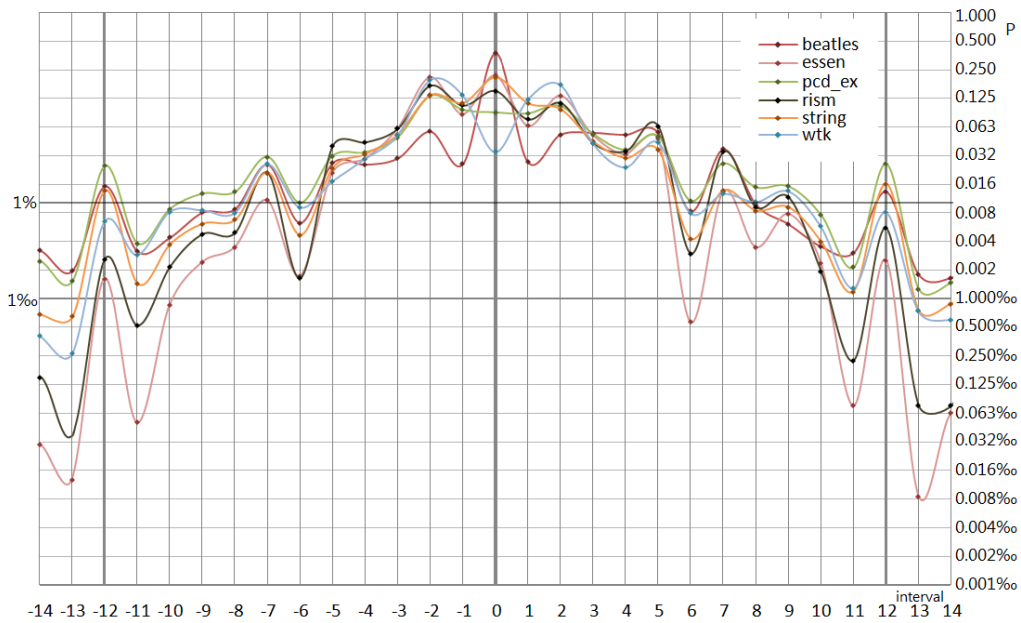
certain intervals have a very specific musical meaning, the numbers they represent, as a difference in MIDI pitches (see Equation 2.9), may not conform to mathematical and statistical laws, so it would be interesting to see, if there are any musically meaningful regularities in the distribution. The same applies to IORs.

Typically, values of intervals and IORs are plotted in linear space [33,41]. However, since most of the symbols occur infrequently, and there is typically a small number of frequently occurring symbols, this indicates that the frequency of symbols may fit some power law distribution. For this reason, we decided to plot values of probabilities in logarithmic scale. Since both melodic intervals and IORs are already defined in logarithmic dependency with basic physical sound parameters, e.g., frequency and time, no logarithmic scaling is applied to those values, unlike in Doraisamy [33], where logarithmic scaling have been applied to IOR values.

Figure 3.2 contains distributions of melodic intervals in all corpora. As we can see in Figure 3.2(a), the probability fluctuates typically in the range up to 10 to 20 times between consecutive intervals, which should be attributed to specific music functions some intervals carry (e.g., multiples of octaves, perfect fourths or perfect fifths). However, despite from rather significant local variations, it is common for all corpora to hold the power law for general trends of probabilities of interval occurrences across the entire interval spectrum. Figure 3.2(b) focuses on its most important part, which contains all intervals within one octave (± 12 semitones), and where musicological dependencies are most apparent. All corpora have rather symmetric and similar distribution of intervals in this range, with several noticeable differences. **beatles** dataset has significantly more unisons ($mi = 0$), than other corpora and significantly less seconds ($mi = \pm 1$ and $mi = \pm 2$), which indicates much less scale passages, and much more notes repetitions comparing to the other corpora, especially **wtk**. Second, is low probability of strictly dissonance intervals (tritones: $mi = \pm 6$, major sevenths: $mi = \pm 11$ and minor ninths: $mi = \pm 13$) in **essen** and **rism** datasets, containing solely monophonic themes and leading melodies and no accompaniment material.

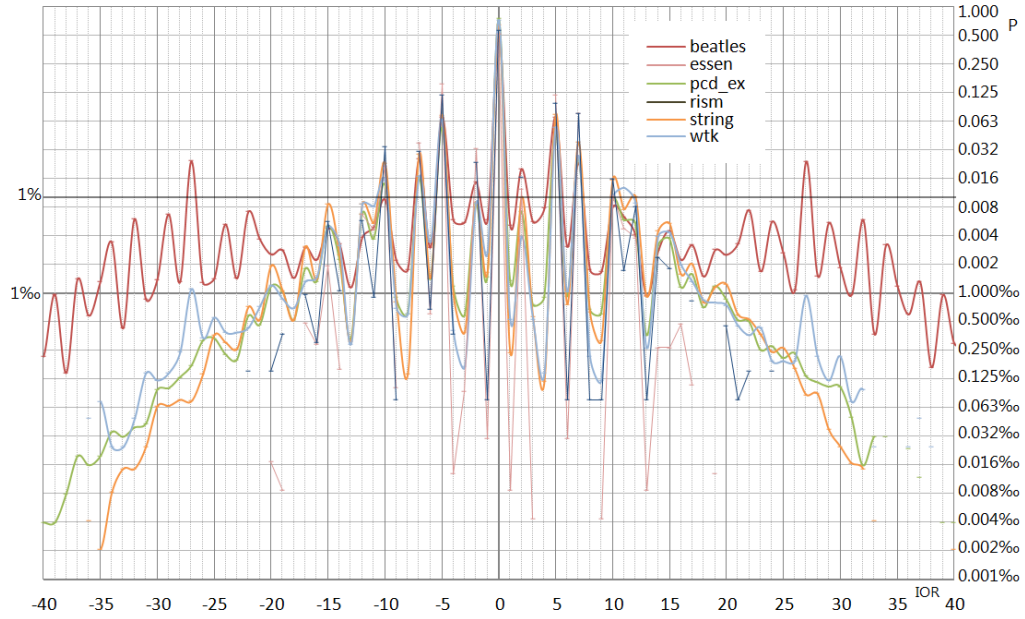


(a) Interval distribution

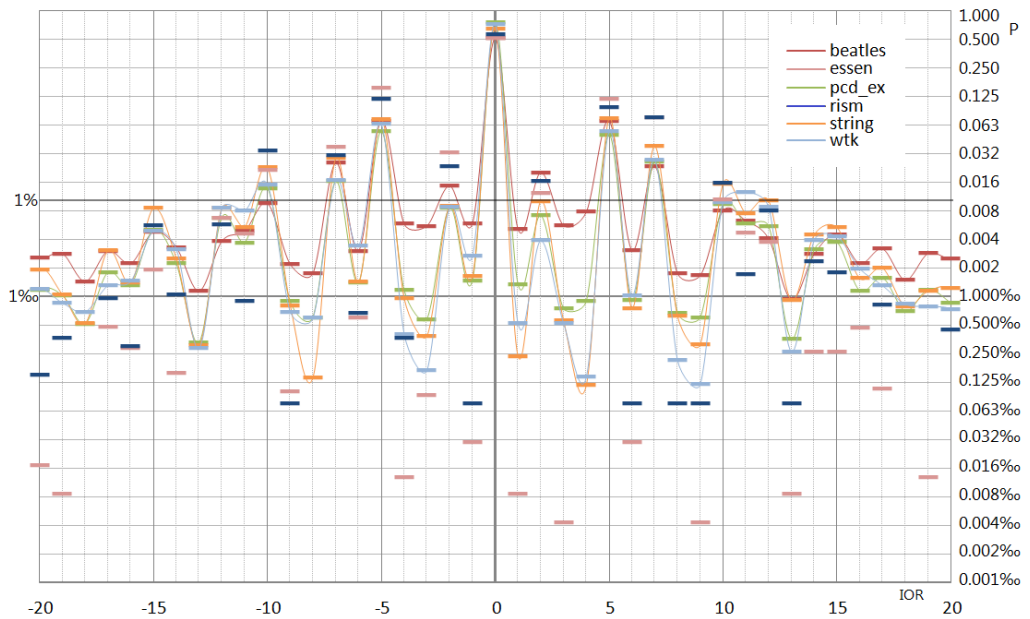


(b) Intervals within 9th

Figure 3.2: Distribution of melodic intervals in the analyzed corpora. a) entire intervals spectrum, b) details of the distribution of intervals within major 9th (scope limited to ± 14 semitones)



(a) Interval distribution



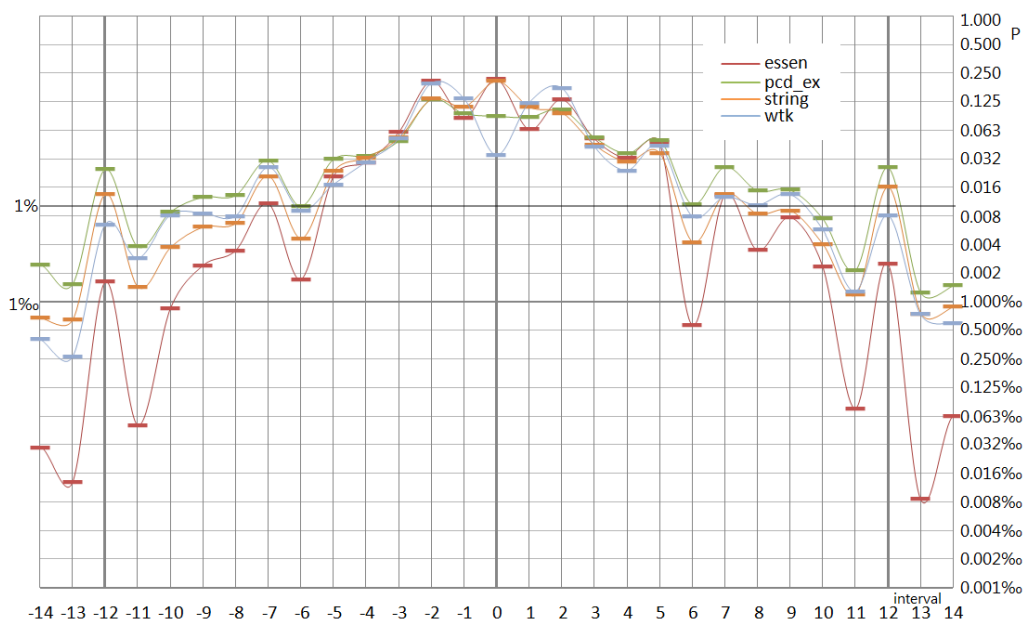
(b) Interval distribution

Figure 3.3: Distribution of rhythmic IORs in all analyzed corpora. a) entire IOR spectrum, b) details of the distribution of IORs within the range of -20 (equiv. duration change of 1/16) to 20 (equiv. duration change of 16/1)

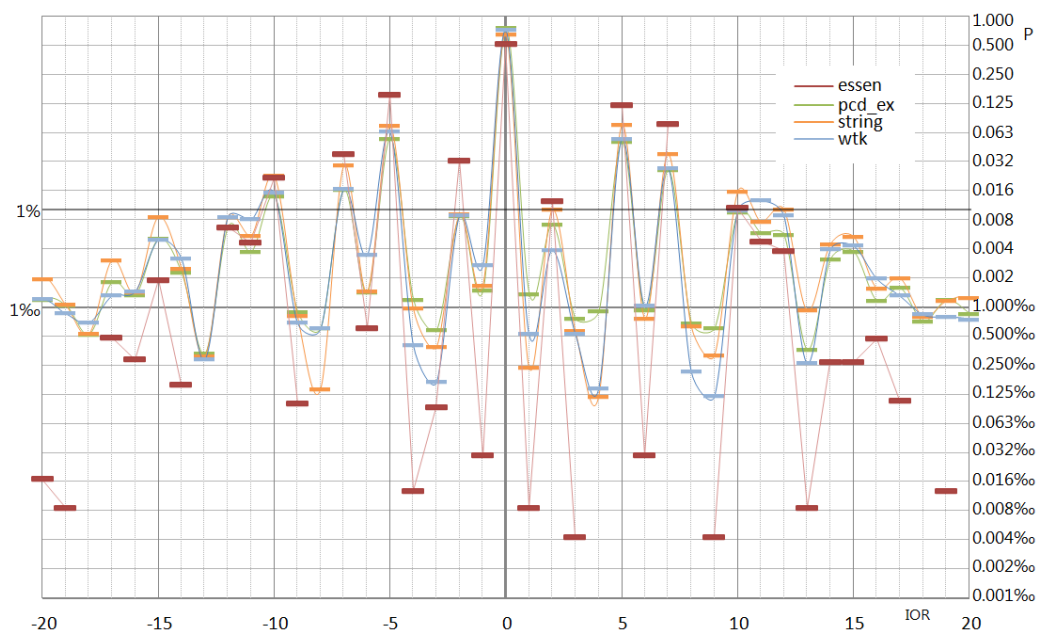
Figure 3.3 contains distributions of inter-onset interval ratios (IORs) as defined by Equation 2.18. Again, the overview of the distributions (Figure 3.3(a)) shows great symmetry with respect to $ior = 0$, and that despite significant local variations, the general trends among most corpora satisfy power law in terms of distribution of IORs, especially in the range of $|ior| \geq 15$. The only exception seems to be **beatles** corpus, where the probabilities above $|ior| = 15$ seem to be accidental, with probabilities between 1% and 1‰. It can be explained by the fact that other corpora do not typically have pieces with tracks for instruments that enter at random times to play for a couple of notes, which is the case for **beatles** corpus, where those random entries mask intentional IORs above 15. Other than that, in the range of $|ior| \leq 15$, i.e., for duration changes less extreme than ♯/♩ or ♩/♯ transitions, typical duration ratios, like 1/1 ($ior = 0$), 2/1 ($ior = 5$), 3/1 ($ior = 7$), 3/2 ($ior = 2$) or 4/1 ($ior = 10$) dominate, which is most pronounced for both incipit datasets (**essen** and **rism**) and to the lesser extent for **beatles** dataset, with the highest number of off-chart rhythmic transitions.

The closer look at four classical music datasets (**essen**, **pcd_ex**, **string** and **wtk**, see Figure 3.4) highlights the lower number of unisons in **wtk** as well as large jumps in **essen** comparing to other datasets. Other general, common for all datasets observations include twice as smaller number of jumps of 4^{ths} down than up, as well as 50% more popular downward passages of 2^{nds}, comparing to the upward direction. Rhythmically, **wtk**, **string** and **pcd_ex** are very similar, with **essen** having typically 10 to 30 times less IORs corresponding to irregular, and twice as more to typical combination ratios.

Plotting corpora containing only melodies and themes, i.e., **essen** and **rism**, on one chart (Figure 3.5) indicates how similar they are with regards to distribution of melodic intervals and IORs. In both cases, we can observe much higher content of jumps up (with $mi \geq 5$), than equivalent jumps down, compensated with more passages in downward direction with $-3 \leq mi < 0$. Rhythmic landscapes are also very similar (Figure 3.5(b)) with very low content of irregular rhythmic changes.

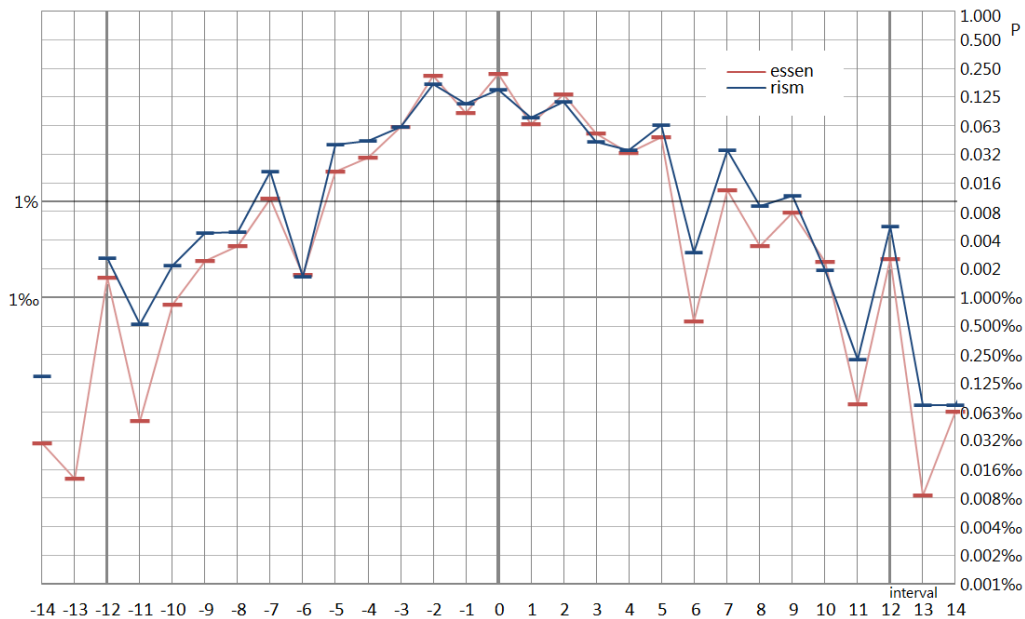


(a) Interval distribution

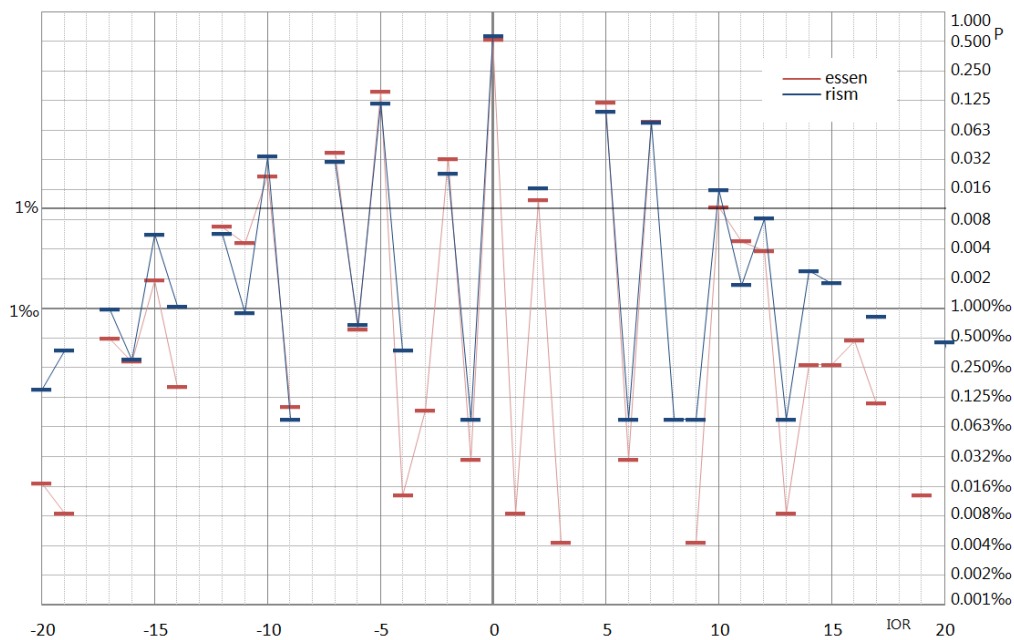


(b) IOR distribution

Figure 3.4: Comparison of distribution of melodic intervals and IORs in classical datasets (*essen*, *pcd_ex*, *string*, and *wtk*): a) melodic intervals with major 9th, b) IORs

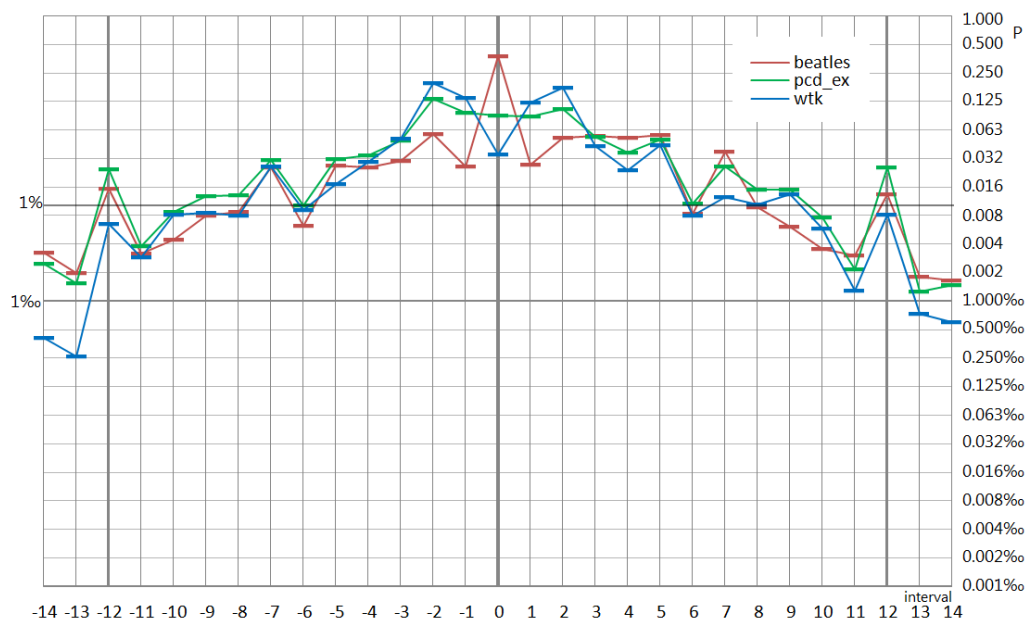


(a) Interval distribution

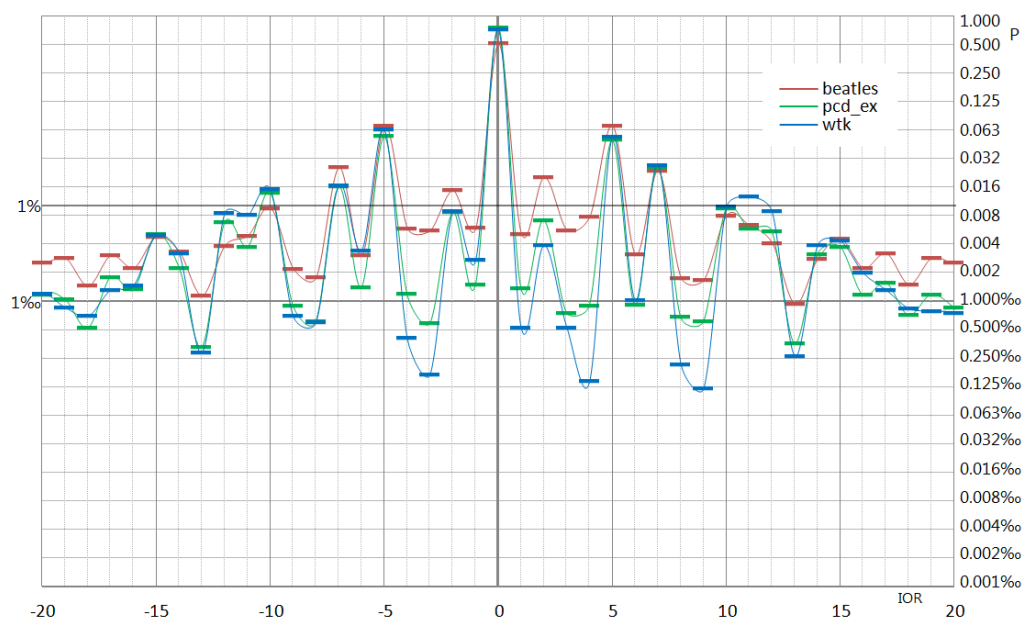


(b) IOR distribution

Figure 3.5: Comparison of distribution of melodic intervals and IORs in MIREX SMS datasets (*essen*, *rism*): a) melodic intervals with major 9th, b) IORs



(a) Interval distribution



(b) IOR distribution

Figure 3.6: Comparison of distribution of melodic intervals and IORs between opposing datasets (*beatles* and *wtk*) with a reference of *pcd* dataset: a) melodic intervals with major 9th, b) IORs

Figure 3.6 shows the difference between `beatles`, containing many more repetitions than melodic passages, and `wtk`, with dominance of melodic passages. Although, the probability of intervals $|mi| \leq 2$ differ are as much as 10 times for those datasets, and these small intervals contribute overall to 50% – 70% of all interval occurrences, it does not affect the interval spectrum for intervals larger than $|mi| \geq 3$, typically with `pcd_ex`, plotted for reference, lying in between them. Similar properties can be seen on rhythmic graph with much higher content of irregular rhythmic changes in `beatles` and much lower in `wtk` (Figure 3.6(b)).

3.4.3 Zipf’s Law for n-grams

George Kingsley Zipf in the book “Human Behaviour and the Principle of the Least Effort” points out that the least effort rule is fundamental to humans and that we may find the manifestations of this law in all human actions [220]. One of the characteristic features of natural languages which, according to Zipf, results directly from this principle, is a very specific distribution of words that we can find in all texts. This can be summarized in one equation:

$$r \times f \approx Const.$$

where r is the rank, or position, of a word in sorted by counts list of all words, and f is the frequency, or count of this word. For every word, the product of its rank r and count f should remain constant, which means that if we plot the frequency and the rank of each word from the corpus, ordered by frequency, on a doubly logarithmic chart, they would form a straight line with a slope of -1 . At the time of Zipf, this was an important and not easy finding keeping in mind that there was no computers to do the counting or sorting, and the entire work was performed manually.

Following Zipf’s reasoning, if the assumption that music is also a natural language, but expressing different things than regular natural languages, that emerged from the same kind of process in human brains satisfying Zipf’s unification—diversification balance [220], we could expect music presenting the same kind of statistical properties.

Downie [41] compared distribution of characters and melodic intervals, however they focused on the most frequent symbols, where usually rank-probability graph fluctuates and does not hold Zipf's law very well. However, Uitdenbogerd [194] analyzed melodic features in the way suggested by Zipf, using the entire corpus of many different n-gram features on a double logarithm graph, achieving results similar to those obtained using text character n-grams (see Figure 3.7(j) or 3.7(i)).

Figure 3.7 contains a number of analyses of terms distribution in music in text corpora. Music terms are n-grams of features combining melodic intervals and IORs. Such tuples contain all the melodic and rhythmic information required to reconstruct the original excerpt. For text, character n-grams and words are being used. Uitdenbogerd in her analysis of n-grams distribution [194] used melodic contour features, which gives a very limited feature space (there is only basic symbols available), which lead to conclusion, that music n-grams do not follow Zipf's power law distribution, but rather form a cap-shape curve on double logarithmic graph, as one can observe on text character n-grams distribution on Figure 3.7(j). The reason of this situation is probably a very limited feature space of basic n-gram components with the use of melodic contour features, similarly to English language from `wiki`, where it uses only a very limited set of characters. Combining melodic intervals and rhythmic IORs creates virtually unlimited number of possible uni-grams combinations, which is the case with English words as well as Thai letters. Apparently, the behaviour of Thai, which is definitely a natural language, is much more similar, in terms of n-gram distribution, to music n-grams obtained from `string` or `pcd.ex` than to English character n-grams from `wiki` dataset.

What looks interesting is the influence of randomization procedure applied in `wiki_rand` and `wtk_rand` corpora. Surprisingly, it does not affect the Zipf's law for those artificially created text words of `wiki_rand` in Figure 3.7(k), which means, that to estimate Zipf's word distribution, obtaining character statistics is sufficient, without the need of counting the actual words. However, what changes after applying randomization, is the behaviour of character n-gram distribution in both datasets.

The original datasets (i.e., `wtk` and `wiki`) keep lines for consecutive values of n close to each other, indicating how little changes in the distributions while increasing or decreasing n . After randomization, the gaps between curves increases significantly, setting them further apart.

3.4.4 Entropy Analysis

Entropy is the measure of uncertainty associated with a random variable. It can be interpreted as the shortest average length of a message, that is required to communicate a true value of the random variable to the recipient, therefore it is considered as a measure of information.

General formula for entropy of a discrete random variable X , which takes values x_1, \dots, x_n with probabilities $p(x_1), \dots, p(x_n)$ is as follows:

$$H(X) = \sum_{i=1}^n p(x_i) \log_b \left(\frac{1}{p(x_i)} \right) = - \sum_{i=1}^n p(x_i) \log_b p(x_i) \quad (3.3)$$

where b determines the unit of information, with the most commonly used bit (with $b = 2$), but there are also situations where nat ($b = e$) or dit ($b = 10$) are used. Whenever k does not occur in a random variable X , it is assumed that the probability of k is 0 ($p(k) = 0$) and its contribution to $H(X)$ is none:

$$\lim_{p \rightarrow 0^+} p \log p = 0 \quad (3.4)$$

Downie [41] calculated entropy of melodic intervals in a corpus of 10,000 folksongs, mostly consisting of Essen Folksongs Collection, at the level of 3.39 bits per melodic interval, which is in line with estimations based on `essen` dataset: 3.36 bits (see Table 3.2) and compared this value to entropy of English text, 4.14 (which varies from the value obtained from `wiki`, because we add numbers to the character set for entropy calculation). We can observe significant differences in music data entropy values among styles and corpora, which result from different interval and IOR distributions, discussed earlier, however Thai and English characters also carry different entropy values. Despite the difference, there are certain meeting points between music and

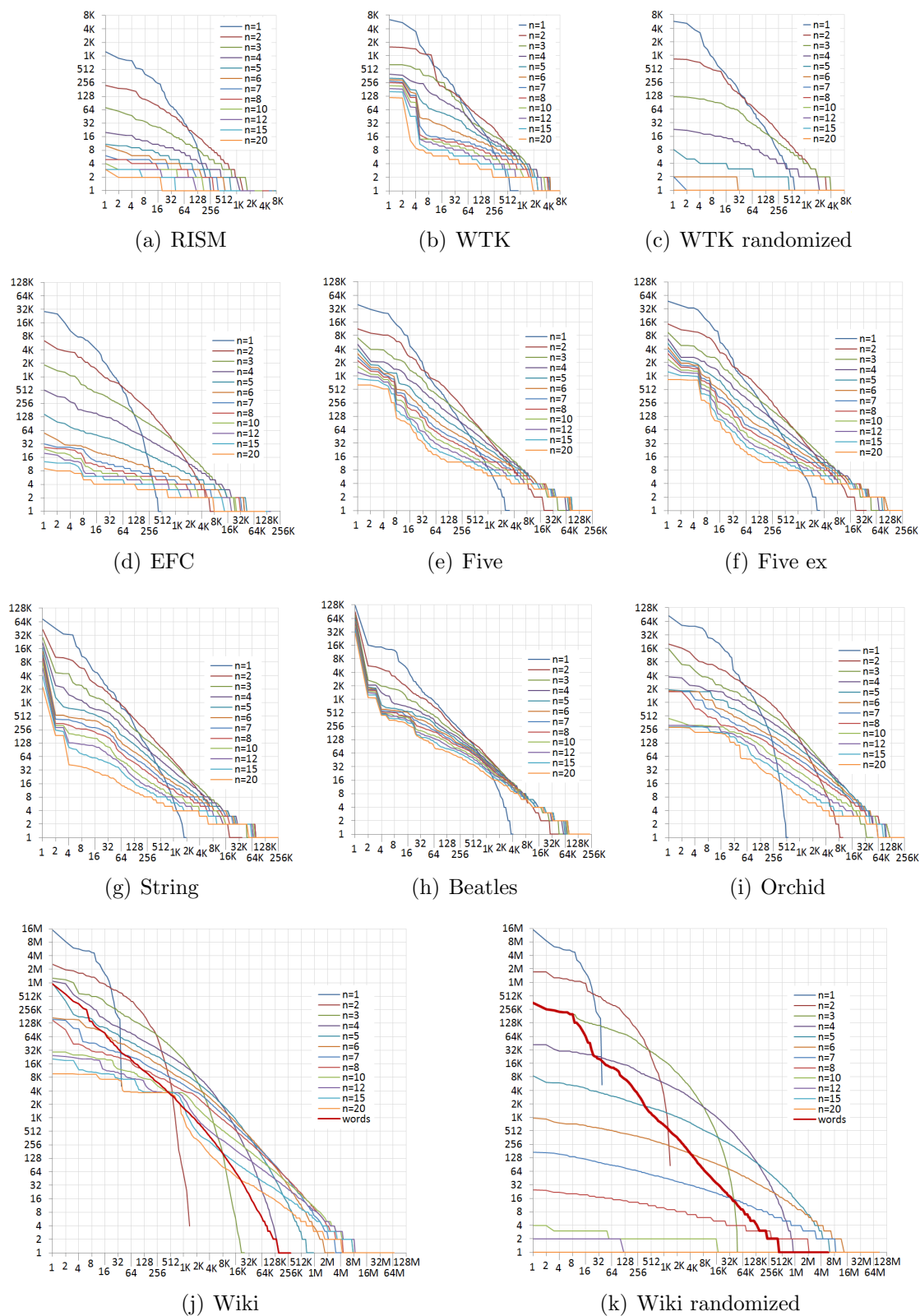


Figure 3.7: Zipf distribution of text character n -grams, music n -grams and text words for analyzed corpora.

Corpus	combined	MI	IOR	characters	words
beatles	7.10	3.89	3.50	—	—
essen	5.68	3.36	2.36	—	—
pcd	6.27	4.40	1.93	—	—
pcd_ex	6.28	4.40	1.93	—	—
rism	5.98	3.72	2.33	—	—
string	6.02	3.79	2.27	—	—
wtk	5.59	3.71	1.99	—	—
orchid	—	—	—	5.99	—
wiki	—	—	—	4.30	11.42

Table 3.2: Values of entropy levels for various corpora, with different feature extraction methods: MI (melodic intervals), IOR (inter-onset interval ratios), combined (melodic intervals with IOR values), and letters and words for textual corpora.

text corpora, e.g., English characters entropy is similar to melodic intervals of `pcd` corpus, and Thai letters entropy corresponds to `string` and `rism` combined (intervals + IOR) entropy values. One can also notice stability of entropy values within certain style, with `pcd_ex` corpus being 25% larger than `pcd`, having exactly the same entropy values.

Entropy of English letters and words were calculated as 4.30 and 11.42 respectively, however with music — there are no explicitly defined phrase boundaries, so one cannot provide a single number as an answer of whatever the entropy of music 'words' is. What can be done is to calculate entropy values for n-grams of different lengths for both music and text datasets. Figure 3.8 contains results of the analysis how much entropy changes with different n-gram lengths. Typically, entropy values increase steady up to the point, where entropy approaches the plateau, limited by the corpus size. Some corpora with shorter documents, especially `rism` and `essen` decrease their entropy for large n , because the documents start to get shorter than analyzed n-gram length, thus failing to provide enough long n-grams. Figure 3.8(a) contains entropies of music corpora with combined melodic intervals and IORs as 'letters' — one can observe that again, `orchid` values are closer to music corpora, than to `wiki`.

Figure 3.8(b) contains analysis of two randomly generated corpora along with their source counterparts. Although they start from different points, slope and the

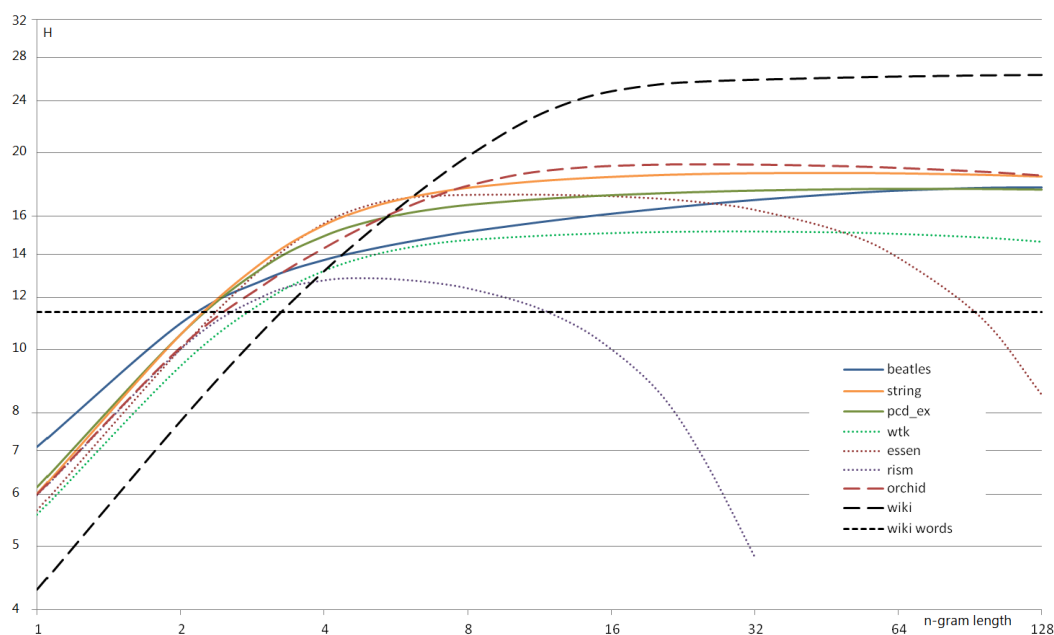
shape of the curves are very similar, indicating that the increase of information as we increase n-gram length is similar. Figure 3.8(c) contains two music corpora with the biggest difference between melodic and rhythmic entropies. Again, although they reach plateau in different manners, the overall behaviour is similar. What is worth noticing, is that not only entropies of melodic intervals from `pcd_ex` and characters from `wiki` are similar for $n = 1$, but as n increases, they stay the same, until `pcd_ex` saturates at $n = 6$.

3.4.5 Complexity and Relations Between Datasets

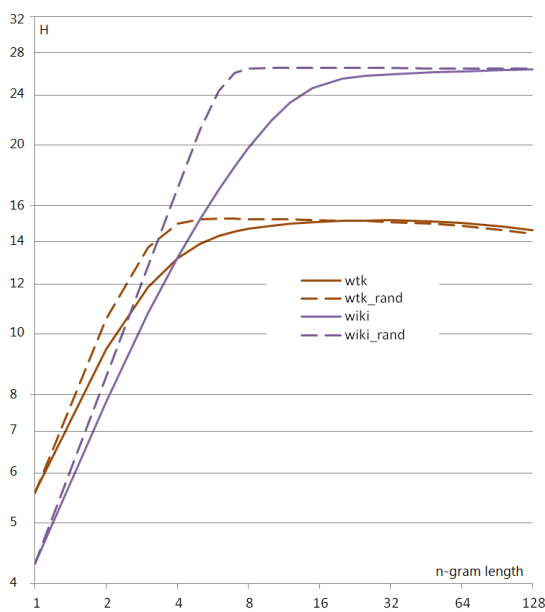
To measure the complexity of data, the notion of Kolmogorov complexity is often used. Briefly, the complexity K of the dataset D , denoted as $K(D)$, is the length of the minimal program that generates D without any input. The question is what programming language should one use to measure Kolmogorov complexity, since typically to find the actual $K(D)$ involves deep (human) understanding of the underlying data type, which is often not desired, as this does not lead to robust solutions. Typically, the upper bound of Kolmogorov complexity is estimated using compression algorithms.

The concept of Kolmogorov complexity is often found in analyses of various types of data, not excluding symbolic music. Approaches available in the literature, either use general purpose compression engines, like `bzlib` [76], or use clean, simple algorithms just to estimate the size of compressor output [114]. Although using industry-grade compression programs usually leads to better compression ratios, which in a sense brings the result closer to the actual Kolmogorov complexity of underlying data, it has its drawbacks:

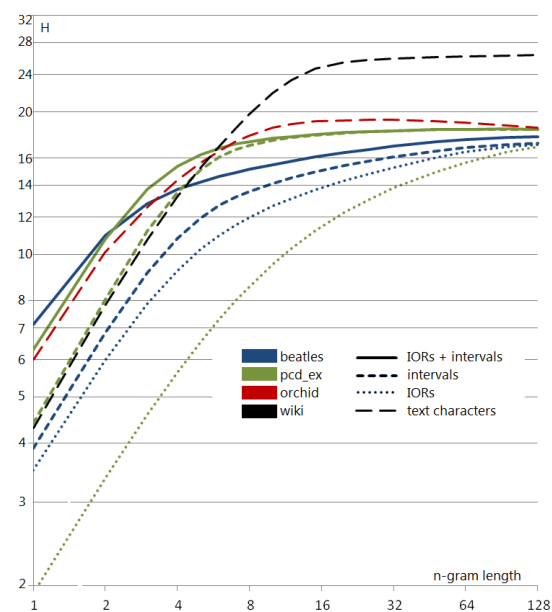
1. The researcher does not have full control and full insight of the actual compression process,
2. The algorithm may be optimized to operate on certain types of frequently occurring data, and



(a) Entropy of various corpora



(b) Randomized corpora



(c) Melodic and rhythmic components

Figure 3.8: Factors influencing entropy estimation: a) Entropy of n-grams of various corpora with different n-gram lengths. b) Influence of randomization on entropy values. c) Entropy of melodic (intervals) and rhythmic (IORs) aspects of n-gram corpora.

3. Algorithms typically suffer from trade-offs between resources used for compression, compression speed and compression ratio.

Using clean, simple algorithm does not have these flaws, and the resulting approach is more theoretically sound, therefore this approach has been used in our analysis.

As suggested in Li and Sleep [114], `lz78` algorithm will be used to calculate $K(D)$. It passes over a sequence of symbols (in this case a sequence of melodic intervals, IORs or intervals and IOR combined), and builds up a dictionary of common past subsequences. In terms of n-gram approach, it can be seen as creating a variable length n-gram profile of the data it is analysing. As the dictionary is the only thing required to reconstruct the initial sequence, the size of the dictionary can be used as an estimation of $K(D)$. For further details on how to compute the size of the dictionary using `lz78` algorithm, please refer to Li and Sweet [114].

Since sizes of the datasets vary and $K(D)$ depends on the size of D , to (partially) solve the problem of comparing datasets with different sizes, we propose to use the ratio between $K(D)$ and the size of D as the measure of Compressibility (C):

$$C(D) = \frac{K(D)}{|D|} \quad (3.5)$$

although typically compression algorithms achieve better ratios when run over larger data streams. Figure 3.9 contains compressibility ratios for all analyzed datasets with separate estimation of compressibility of rhythmic IORs, melodic intervals and combined features (intervals and IORs) for music dataset. Text datasets were interpreted as streams of characters. One can notice higher compressibility of text datasets, but this can be attributed to larger sizes of those corpora. An interesting observation can be made if one overlays entropy information with compressibility ratios. Relative entropy levels between melodic intervals and IORs correspond to compressibility obtained for a given dataset. Once again, `beatles` dataset seems to have the richest rhythmic spectrum comparing to other datasets. `rism` and `wtk` datasets achieved the worst compressibility scores (the highest in values) due to their small size. One can also notice worse compressibility of randomized corpora (`wtk_rand` and `wiki_rand`)

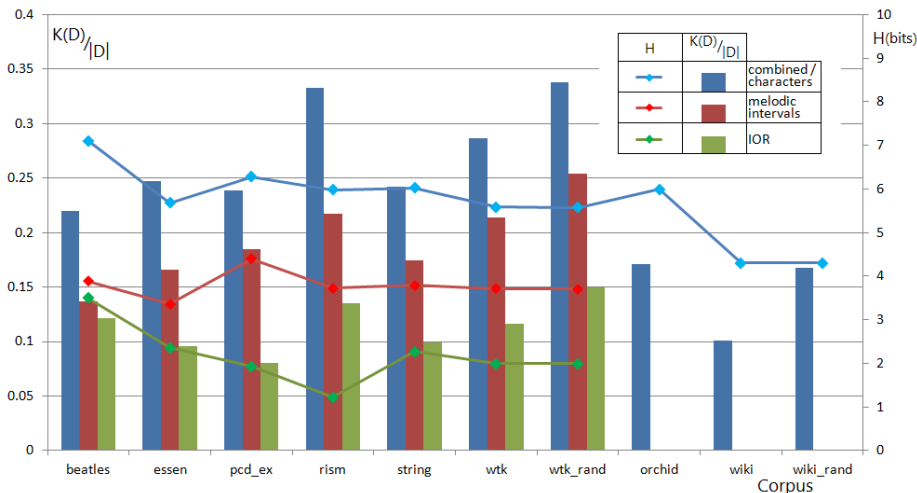


Figure 3.9: Compressibility of various datasets for combined melodic interval and IORs, only melodic intervals and only IORs along with compressibility of text datasets. For comparison, the entropies of corresponding corpora (H) are added.

comparing to their base counterparts, despite the same size and symbols entropy, indicating the importance of order of symbols in both music and text datasets.

As mentioned before Kolmogorov complexity $K(D)$ is the length of the shortest program to generate D without any input. Conditional Kolmogorov complexity, $K(D|T)$ is defined as the length of the shortest program to generate D with T available at the input. Likewise, joined Kolmogorov complexity, $K(TD)$ is the length of the program to generate T and then D without any input. According to Li and Sweet [114], one can define $K(TD)$ as the complexity of concatenated streams of T and D :

$$K(TD) = K(T.D) \quad (3.6)$$

and conditional complexity $K(D|T)$ as:

$$K(D|T) = K(TD) - K(T) \quad (3.7)$$

Li and Sweet [113] proposed two Normalized Information Distance metrics, d_1^K and d_2^K , based on the notion of conditional Kolmogorov complexity, which can be expressed as similarity measures with the following equations:

essen	2.07					
pcd_ex	2.73	2.77				
rism	0.91	2.05	1.09			
string	2.65	3.28	4.40	1.15		
wtk	1.25	2.21	2.52	4.63	1.98	
wtk_rand	0.98	1.47	1.37	2.99	1.31	3.77
d_1^K	beatles	essen	pcd_ex	rism	string	wtk

(a) d_1^K measure

essen	3.27					
pcd_ex	4.95	4.05				
rism	0.94	2.03	1.09			
string	4.76	4.77	8.54	1.15		
wtk	1.41	2.59	2.71	5.91	2.15	
wtk_rand	1.18	1.84	1.56	3.54	1.46	6.70
d_2^K	beatles	essen	pcd_ex	rism	string	wtk

(b) d_2^K measure

Figure 3.10: Similarities between music corpora using d_1^K and d_2^K measures based on Kolmogorov complexity estimation (values in percents)

$$d_1^K(x, y) = 1 - \frac{K(x|y) + K(y|x)}{K(xy)} \quad (3.8)$$

$$d_2^K(x, y) = 1 - \frac{\max(K(x|y), K(y|x))}{\max(K(x), K(y))} \quad (3.9)$$

They operate on the premise that two similar sequences of symbols, when compressed together, would achieve better compression ratio than two less similar sequences. Both measures can be used in place of any similarity measure for tasks where such is required, and since they do not require any parameter tuning (like most n-gram based similarity measures), they can be used for preliminary analysis of similarities between different datasets. Figure 3.10 contains results of such analysis, without significant differences between results of d_1^K and d_2^K measures. What was expected, as a result of randomizing, `wtk_rand` dataset is less similar to all original datasets, comparing to the results obtained for baseline `wtk`. Also, `beatles` dataset, the only contemporary music dataset, achieved generally low similarity with other datasets. What is surprising is the relatively low similarity score between `essen` and `rism` and high score between `wtk` and `rism`.

3.5 Composer Classification of Symbolic Corpora

As it has been shown in this chapter, various symbolic music corpora share many similar properties with natural languages, which supports porting existing text analysis methods to music data. One of the hallmarks of text data mining is classification task, where a program, based on a number of documents assignments to specific classes, has to assign a class label to an unseen object. Accuracy, the ratio of correctly assigned items from the testing set to the size of the testing set, is typically used as a measure of classification performance. Usually, systems that classify text documents employ bag-of-words approach which assumes that the order of terms, or words in the document does not matter and the whole document is perceived as a set of features, typically words.

This model can be applied to symbolic music if one takes n -grams as features. This however, poses a new problem, as since n , the n -gram length, is not given a priori, which creates a new degree of freedom for the system and even non-parametric methods effectively become parametric when applying this model. And, like with all parametric methods, it would require fine tuning of what n should be taken, which involves the risk that the optimal n may change upon conditions that are difficult to foresee.

To evaluate classification performance of various text-derived models on symbolic music, two of our test collections can be used. The `pcd` dataset contains items classified into 5 classes, each representing not only a different composer, but typically different style and usually containing different forms of music. In the previous work, it have been shown, that one can obtain high accuracy scores despite having many classes [216]. What has been missing is the comparison with other approaches and other datasets. This can be achieved with the inclusion of `string` dataset.

The `string` dataset contains two classes of highly similar content, of two different composers, but living in the same place and times, and of the same form (string quartets). Having a very uniform dataset sounds like a good opportunity to set up a human classification challenge. CCARH hosts such a quiz, allowing anybody to

participate and guess who wrote a given piece. With over 40 thousand participations, they have precisely measured human accuracy of 57% (56% for Mozart and 58% for Haydn), although with this very unbalanced dataset, a majority classifier would score 72% pointing solely to Haydn (with 0% for Mozart and 100% for Haydn).

Furthermore, Hillewaere et al. [75] evaluated classification performance of several different methods based on a dataset derived from the original `string` dataset, which we included as `string_rh`. According to them, `string_rh` contains fewer pieces characteristic for Haydn, increasing the difficulty of the problem, and reducing imbalance from 72% for Haydn to 56%.

Evaluating classification performance of a number of classical approaches on all three datasets at the same time would allow comparison of the previous work done on `pcd` dataset, human evaluations of `string` dataset hosted by CCARH, and approaches used by Hillewaere et al. [75] on automatic classification of `string_rh` dataset.

3.5.1 Methodology

Evaluation procedure. Due to small size of datasets in question, we have chosen to rely on leave-one-out N -fold cross-validation for testing. It takes one item from a collection, apply training based on the remaining $N-1$ items and repeat the procedure for each item in the collection, aggregating accuracy score over the entire dataset.

Full and balanced training. Due to imbalanced nature of all three datasets, we decided to test performance of the algorithms in two variants. The first one is the unchanged cross-validation procedure where the entire dataset is available for training. For the second one, the data supplied for each testing algorithm is limited to N_{max} which is the maximum possible number of tokens allowed for all items to be tested in the cross-validation procedure:

$$N_{max} = \min_{class \in Classes} \left(\sum_{document \in class} (|document|) - \max_{document \in class} (|document|) \right) \quad (3.10)$$

Forcing balancing of training datasets reduces some training information for most classes, but removes the bias in the results that arise from comparing sets of different lengths or numbers. As the outcome, the only bias seen in the results with balanced training comes from the unique features of n-gram landscapes of a particular class.

N-gram lengths. With text corpora and bag-of-terms approach used on words, this problem does not exist. There are however situations, where text character n-grams are used [93], or language simply does not provide word boundaries (see `orchid` corpus), and the issue, what is the proper value for n arises. While typically for symbolic music small values of n-gram lengths are suggested (up to $n = 5$), to test the broader landscape, we tested for more values, up to $n = 15$. This allowed us to observe how, and if, the performance converges when long n-grams are being used.

Importance of rhythmic and melodic dimension. With text, there is no problem on how to interpret particular characters from the input. Since music features are multi-dimensional (i.e., typically are placed in 2-dimensional melodic-rhythmic space), it is important to test individually what is the usefulness of particular aspects of symbolic music spectrum. Here, melodic intervals and inter-onset interval ratios (IORs) are being tested, both separately and combined to form a single dimension joining melodic and rhythmic aspects. This combined feature is particularly interesting, as while still being transposition and tempo invariant, they keep full information about the melody, allowing the original melody to be fully recovered.

Similarity algorithms. The core of the classification procedure is the similarity (or distance) function which compares an individual with the rest of the training data. The similarity functions evaluated in this experiment are the following:

1. **common** — As suggested by Suyoto and Uitdenbogerd [175], this should be the simplest, baseline method of comparing two feature sets, which returns a number of features in common as a measure of similarity:

$$common(X, C) = |X \cap C| \quad (3.11)$$

where X is the individual being compared to the set representing all features from class C . Dice coefficient and Jaccard index (introduced later in the dissertation, section 6.3.1) can be seen as two extensions of this measure, with two different normalization approaches, while here no normalization is applied. Evaluating it would allow one to observe, whether other more complicated functions introduced below, benefit from their sophistication.

2. **markov** — Hillewaere et al. [75] used this model to evaluate classification performance on `string_rh` dataset. It uses Markov assumption directly, calculating probability of the measured document as the product of probabilities of each symbol that depend only on the previous $n - 1$ symbols:

$$P(X|C) = \prod_{x_i \in X} p(x_i | x_{i-1}, \dots, x_{i-n+1}) \quad (3.12)$$

where probabilities $p(x_i | \dots)$ are drawn from C class n -grams statistics. Add-one smoothing have been applied to the probabilities to accommodate for potential zero-probability events that would zero the final product. In the original approach, Hillewaere et al. [75] fixed the parameters for the system to $n = 3$ and melodic intervals as features, so it would be interesting to observe how this method performs in broader contexts.

3. **cosine** — It is classical cosine similarity, which is the cosine of the angle between vectors representing X and C in multi-dimensional n -gram feature space:

$$\text{cosine}(X, C) = \frac{\sum_{i \in X} X_i C_i}{\sqrt{\sum_{i \in X} X_i^2} \sqrt{\sum_{i \in C} C_i^2}} \quad (3.13)$$

As typically used for text comparison, it is relevant to include it in this analysis.

4. **cng** — A measure used in previous work [216], to evaluate classification performance on `pcd` dataset. Here, the non-parametric version have been used, although in [216], there is a thorough analysis of influence of additional factors, like profile sizes limits and the ageing factor, which improve results obtained using plain **cng** function. In essence, **cng** measure uses normalized arithmetic

differences of probabilities of n-grams in both compared objects:

$$cng(X, C) = \sum_{i \in X \cup C} \left(1 - \left(\frac{X_i - C_i}{X_i + C_i} \right)^2 \right) \quad (3.14)$$

A different, not normalized version, which can be found in Ukkonen [199] has been used with music n-grams by Uitdenbogerd and Zobel [193], and Mullen-siefen and Frieler [132].

5. **lz** — It uses LZ78 compression algorithm to estimate conditional complexity of a document X given class C . Since it can be used as a measure of distance (i.e., the more similar objects, the smaller the value), we used a simple negation to obtain similarity measure:

$$lz(X, C) = -K(X|C) = K(X) - K(CX) \quad (3.15)$$

Since complexity K operates on individual melodic and/or rhythmic features, and builds a dictionary of variable length n-grams while scanning the data stream, it does not depend on parameter n .

Precision of measuring accuracy. Every time a single evaluation is performed, the data is available to the algorithm in a random order, which makes a difference for some algorithms. What is more important, with forced balancing on training data, every time an algorithm receives different data from a number of random documents. This creates variability of the obtained accuracy scores for each performed evaluation. To accommodate for that, each test is performed many times, until the standard deviation of the mean accuracy estimation falls below 1%. Initially, every algorithm with certain settings is evaluated 10 times. If it does not return consistent results (with $\sigma > 0.03$, hence $\sigma_{\text{mean}} > 0.01$), it is repeated until desired $\sigma_{\text{mean}} \leq 0.01$ is obtained. Results are posted with error bars indicating the dispersion (standard deviation) of results for given settings, which is not the error of average performance estimation (standard deviation of the mean), which is kept, as mentioned, below 1% through exhaustive evaluation.

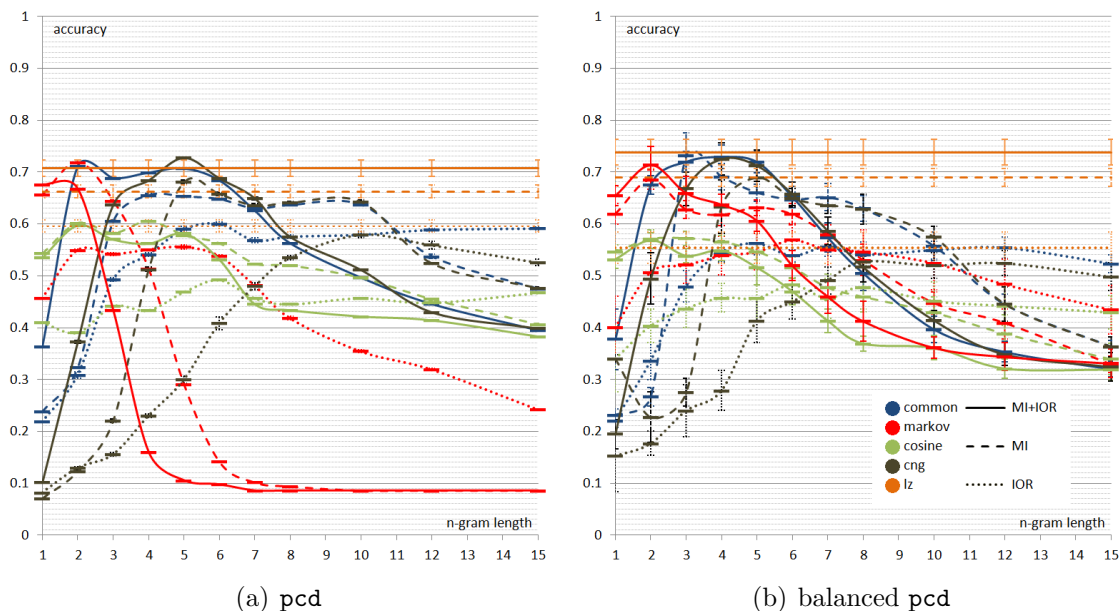


Figure 3.11: Accuracy of classification for pcd dataset with: a) full training data, b) balanced training data

3.5.2 Results

Overall results for pcd dataset. The highest accuracy scores with full training have been obtained for $2 \leq n \leq 5$, depending on the method, with similar scores (around 72% scored by `markov`, `common` and `cng` methods (Figure 3.11). `lz` algorithm scored 71%, while `cosine` fall significantly below others. Previous work [215,216] focused on analysis of various parameters associated with `cng` measure, being able to put the results up to 80%, however, due to lack of analysis of other datasets, the obtained high accuracy scores may not transfer well to other datasets and settings. Introduction of balanced training increases performance of most top performing algorithms, being able to push the results to around 74% (see Figure 3.11(b)).

Overall results for string dataset. Most top performing algorithms from pcd results fall quickly into majority or minority classification, fixing themselves in pointing consistently to just one class (Figure 3.12(a)). The `string` dataset is highly unbalanced with Haydn pieces — the easiest class to be classified, being also the

most numerous. The best and the most balanced results were obtained, rather surprisingly, for `markov` method with rhythmic features (accuracy about 60% for both classes). However the most consistent and stable (unbiased) results were given by `cosine` measure, with the accuracies ranging from 55% to 60%, regardless of the class (Figure 3.13(a) and 3.13(c)). By the introduction of balanced training, top accuracy scores stay around 70% to 75% (Figure 3.12(c)). However, we do not see anymore the situation where an algorithm would entirely bias toward just one class, which makes those results much more desirable comparing to full training. Top performers were `common` with combined and melodic features and `markov` with melodic features, although `markov` seems to be better at providing much more consistent results for `haydn` and `mozart` class separately (Figure 3.13(b) and 3.13(b)).

Overall results for `string_rh` dataset. The `string_rh` dataset is indeed much more balanced, comparing to `string`, which is clear after looking at algorithms performance on Figure 3.12(b), where most algorithms scored above the baseline 54% of majority classification for this corpus. Best results were obtained with `markov` method for melodic intervals, providing very balanced results. This method have been used by Hillewaere et al. [75] on this dataset, where they used a 3-gram model. According to our analysis, it seems that increasing n to 4 or 5 would yield even better results. Performance of the second best method, `common`, was highly driven by their increased accuracy in `haydn` class. Enforced balanced training improved results of most methods (`common,cng`), which bring them closer to the top `markov` (see Figure 3.12(d)), improving results mostly for `mozart` class and reducing bias, mainly for the `markov` method (see Figure 3.14(b) and 3.14(d)). Top results obtained for with balanced training were on par with SVM results reported by Hillewaere et al. [75].

Results for the `common` similarity measure. Although being a very simple method, based just on calculating the number of common n -grams between documents, `common` similarity measure performs really well on the `pcd` dataset (see Figure

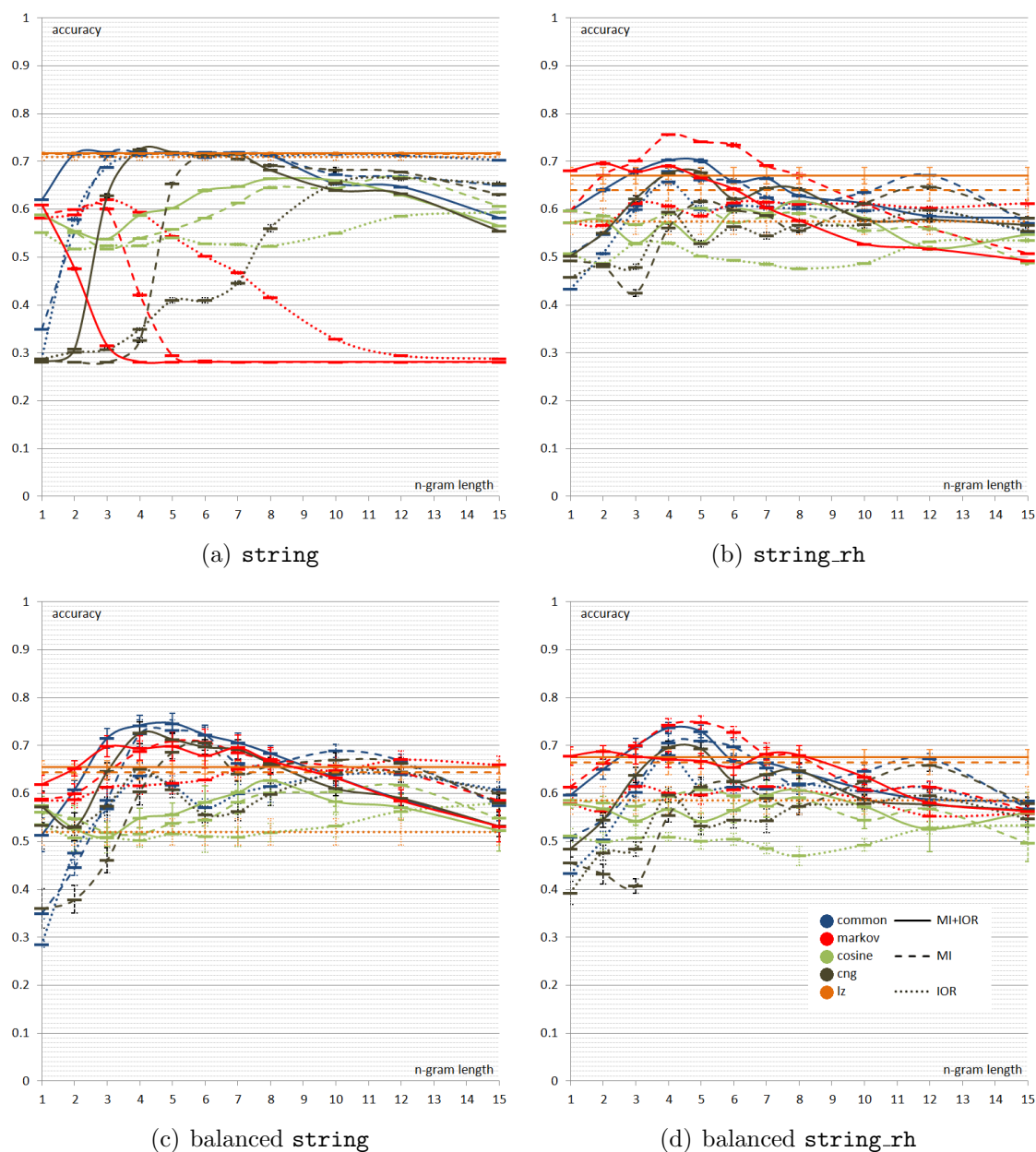


Figure 3.12: Accuracy of classification for `string` and `string_rh` datasets with: a),b) full training data; c),d) balanced training data

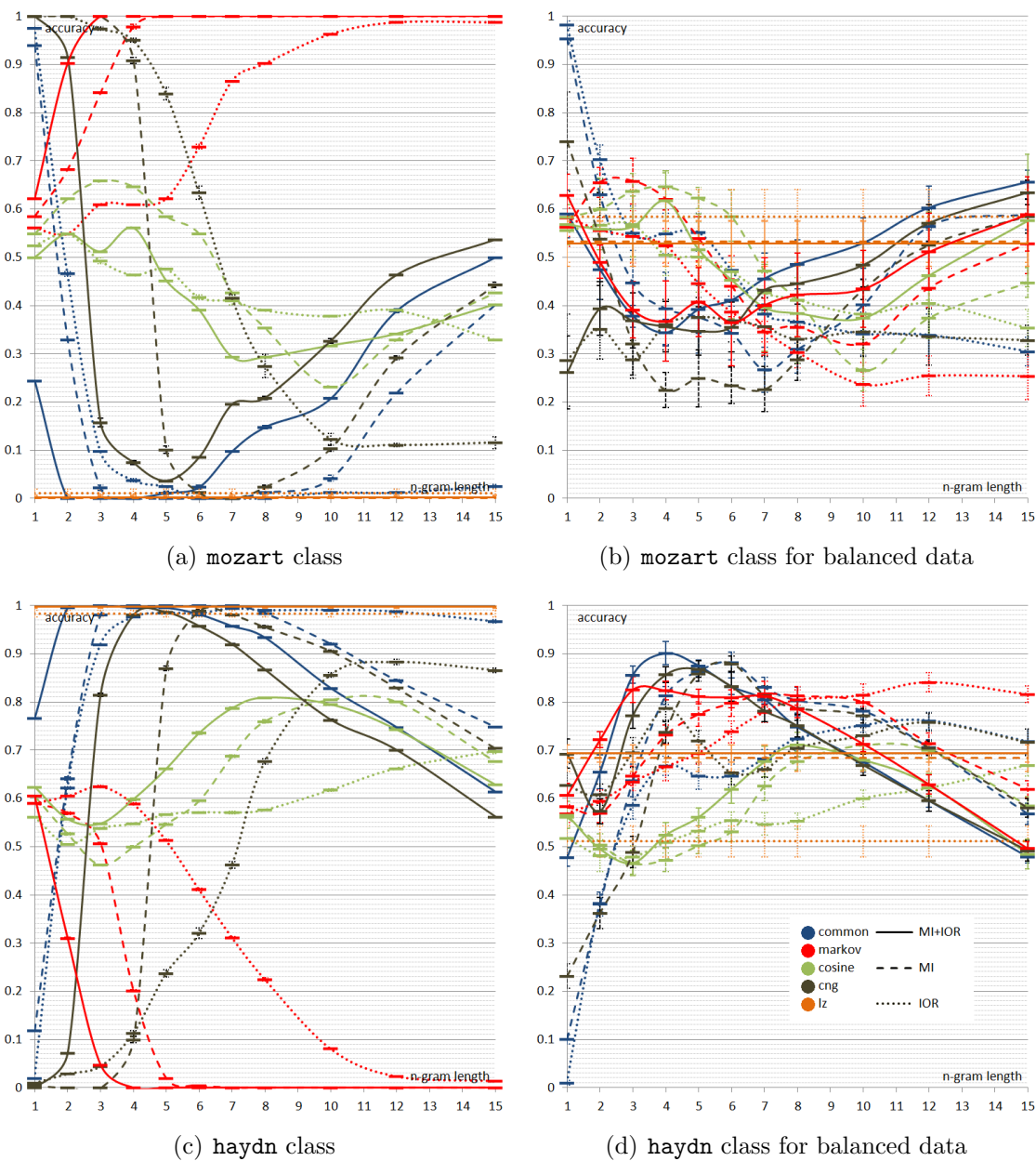


Figure 3.13: Accuracy of 'Mozart' and 'Haydn' classes classification for string dataset. a) mozart class with full training data, b) mozart class with balanced training data, c) haydn class with full training data, d) haydn class with balanced training data.

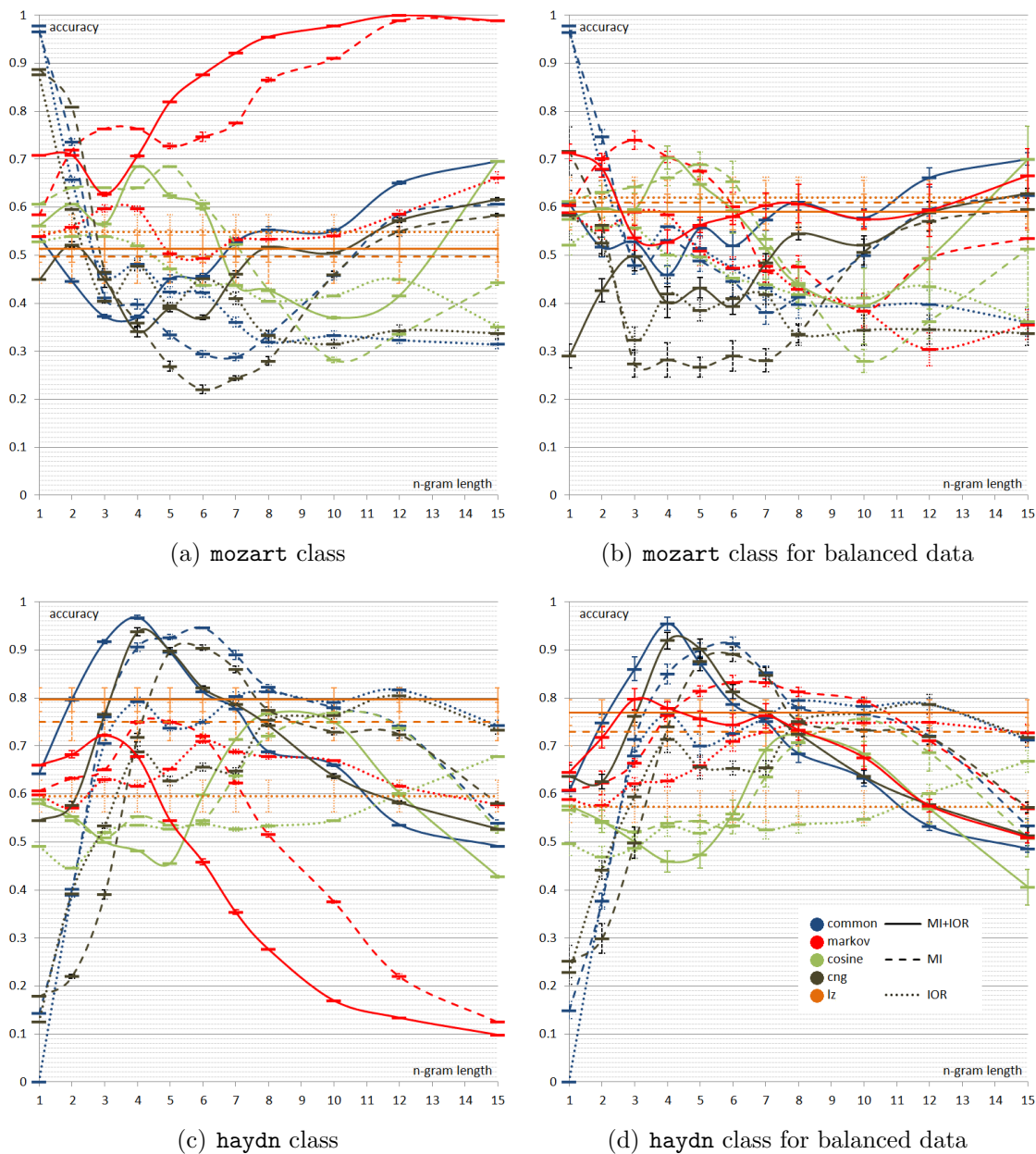


Figure 3.14: Accuracy of 'Mozart' and 'Haydn' classes classification for `string_rh` dataset. a) `mozart` class with full training data, b) `mozart` class with balanced training data, c) `haydn` class with full training data, d) `haydn` class with balanced training data.

3.11), especially with combined features, reaching accuracy of 73% with full and balanced training. Using only melodic intervals as features reduces performance by about 5% overall, and using only rhythmic features reduces it even more, to maximum of 55% accuracy for balanced training. The algorithm has very even performance in its sweet spot, for $2 \leq n \leq 5$, which means, that it should be a robust performer under different conditions and different datasets.

Evaluation on `string` dataset (Figure 3.12(a)) reveals high bias toward majority class, still `common` being one of the top performers (with combined MI and IOR features) for both `string` and `string_rh` datasets for $n = 4$ and $n = 5$. It keeps favouring the `haydn` class even with more balanced `string_rh` dataset and forced balance on training data for both corpora, with average performance for, the harder `mozart` class. What is interesting is that, the majority class bias occurs for $n > 2$, and it favours the `mozart` class for small n , which is not the case for any other similarity algorithm tested in this analysis. Overall, the best results are obtained with balanced training for $n = 4$ with 75% overall accuracy for `string` dataset, with 35% accuracy for `mozart` and 90% accuracy for `haydn`, and 74% overall accuracy for `string_rh` dataset, with 46% accuracy for `mozart` and 95% accuracy for `haydn`.

Results for the markov similarity measure. This measure performs very well for small n-gram lengths ($n < 6$). Above that value it is prone to favouring one class even, if it is not a majority class. Bias is its biggest problem, with the biggest changes between similarity measures noted after introducing forced balanced training, which is most visible for `string_rh` dataset results (see Figures 3.14(a) and 3.14(c)). For small n-grams, it is one of the best performers, yielding the best results, using melodic interval features, for all three datasets, although being very sensitive to changes of n . $n = 3$ and $n = 4$ gave best results for `string_rh` dataset, but with those settings, performance on `pcd` was one of the worst, where the sweet spot was around $n = 2$. Performance for combined melodic intervals and IORs was comparable to just melodic intervals. Using just rhythmic features typically brought results 10 to 20% down, comparing to other features.

What can be observed, especially on `string` dataset results, with smaller n , is that `markov` measures are very resistant to majority class bias, while they lose this resistance quickly with growing n where they bias towards minority class much faster than other methods (see Figures 3.11(a), 3.13(a) and 3.13(c), as well as 3.14(a) and 3.14(c)). Overall, the best results are obtained for $n = 5$ with 71% overall accuracy (54% for the `mozart` and 78% for the `haydn` class) for the `string` dataset and 76% overall accuracy (76% for the `mozart` and 75% for the `haydn` class) for the `string_rh` dataset. The best unbiased results for the `string` dataset were obtained with $n = 3$, $n = 4$ with accuracies for both classes in the range from 65% to 70%. With CCARH estimation of best scores obtained with experienced users around 60%; this is a very good result.

Results for the cosine similarity measure. Cosine similarity was the worst performer in terms of overall accuracy scores, for all datasets, scoring typically 10 to 20% less comparing to the other, top performing methods. On the other hand, it was the least susceptible to bias in the training data, scoring just above 50% for both `mozart` and `haydn` class for both `string` and `string_rh` datasets. Unsurprisingly, the introduction of forced balance in training data did not affect its performance. Combined melodic and rhythmic features outperformed methods using just melodic intervals or just IORs for `string` and `string_rh` datasets. In case of the `pcd` dataset, melodic features performed similarly to combined features, still being significantly above accuracy scores obtained with IORs only.

Results for the cng similarity measure. The `cng` measure achieved the highest scores for `pcd` dataset, obtaining 73% for both balanced and full training. Its preferred n -gram lengths are around $n = 4$ and 5, which, unlike for the `markov` method, is consistent across corpora. Like `common` measure, it turns into a majority classifier for `string` dataset, favouring `haydn` class. It does so, even after applying forced balance in training as well as for the more balanced `string_rh` dataset, being typically one of the worst performing approaches in terms of accuracy in the `mozart` class (see

Figures 3.13(a), 3.13(b), 3.14(a) and 3.14(b)), although the gap between the `haydn` and `mozart` classes diminishes when balanced training is in effect. For the `string` and `string_rh` datasets, achieved accuracy scores are typically a few percent below top performing `markov` and `common`, on par with `lz` and significantly above `cosine`. Combined melodic intervals and IORs perform much better, compared to intervals and IORs alone, with much bigger differences than the one observed with `common` and `markov` methods.

Results for lz similarity measure. The `lz` method achieved one of the best scores for `pcd` dataset (see Figure 3.11), with 70% for full, and 73% for balanced training data for combined melodic intervals and IOR features. It was also a solid performer for `string_rh` dataset. For original `string` dataset, it falls into majority classification, achieving even lower score after applying balanced training, although one can notice that it never falls below 50% accuracy for any class except for highly imbalanced `string` dataset, where all algorithms (except for `cosine`) did not do well. One can notice a significant variation of obtained results (error bars on Figures 3.11(a) and 3.12(b)) even with full training data, due to high sensitivity to the order in which the data is provided for the training. It is not the case for any other algorithm.

Influence of selected n-gram representation method. Typically, results obtained using combined melodic intervals and inter-onset interval ratios achieved better results, in comparison with using only melodic features. Using rhythmic features typically yielded the worst results. It is true for `common`, `cosine`, `lz` and `cng` methods. Melodic intervals used without IORs achieved better results only for the `markov` method for some datasets. Despite that, rhythmic features performed better for larger n-gram lengths, being less prone to bias towards majority classes (see Figure 3.12(a)), so depending on the circumstances, they still may be a preferred over other music features in some cases.

Influence of forced balanced training. Although forcing balance training was introduced primarily to reduce bias associated with imbalanced dataset, it also improved results by several percent for all analyzed datasets. Despite removing physical bias resulting from different size of training data, there is still tendency in `string` and `string_rh` datasets towards `haydn` class, indicating that indeed, as Hillewaere pointed out in [75], Haydn seems to have more characteristic pieces, comparing to Mozart, when considering string quartets.

With better average results comes greater uncertainty of the achieved accuracy scores. Balancing training data allowed for, and required, shuffling and changing data used for training each time an algorithm was evaluated, which had a great influence on the accuracy scores obtained by all similarity measures. However it turned out that the greatest variability in accuracy was observed for average and poorly performing algorithms, leaving no doubts which of the algorithms was the best in each category. Overall, the influence of balanced training was beneficial for the obtained accuracy results.

3.5.3 Summary of Composer Classification Experiment

Our experiments with composer classification task show that many n-gram-based methods, when used with a proper settings, may produce very good results. One of fundamental factors, usually assumed through intuition or music knowledge, is the length of n-gram features used for analysis, which turned out to be a very important and difficult parameter to set, with optimal values varying between similarity measures and datasets. Nevertheless, most methods were able to achieve good results, outperforming humans in Haydn/Mozart string quartets quiz. Typically, the best results were obtained with combined melodic and rhythmic features, and the worst performance with rhythmic features only. Another important issue is imbalance in training data. We have shown, that one way to eliminate it is to limit the training data to equal size portions for all the classes. It turns out that this approach leads not

only to more balanced results, but to better results overall, better than the majority classifier, even when the majority class is dominating, like in the `string` dataset.

3.6 Symbolic Music Similarity

In the following section, we evaluate several factors that influence the performance of n-gram-based music similarity algorithms. Those algorithms are derived from textual information retrieval and adapted to operate on music data. The influence of n-gram length, applied feature extraction method, term weighting approach and similarity measure to the final performance of the similarity measure has been analyzed. The main contribution of this approach is to utilize well established term weighting methods for text retrieval and check their suitability for music data. MIREX 2005 data and the MIREX 2011 evaluation framework for symbolic music similarity task have been used to measure the impact of each of the factors. We have found out that the choice of a proper feature extraction method and n-gram length are more important than the applied similarity measure or term weighting technique.

It has been shown, that bag-of-words methods, which are well-established approaches in textual Information Retrieval, work well for retrieving relevant melodies from music corpora. It is noticeable that the main focus has usually been put on how to transfer the input sequences of notes into a set of features and how to compare (measure the similarity) between different feature sets, yet there has not been sufficient quantitative analysis on how to make proper decisions regarding certain parameters. Usually the parameters are chosen based on music knowledge and intuition rather than experimentation.

It has been found in textual information retrieval, that simply using words as terms and basic similarity measures between terms (like string equality or stemmed string equality) is sufficient. Current research focuses on term weighting, i.e., given documents in a dataset — to determine which terms are more important for each document or the dataset as a whole. Some frequent terms (dubbed stopwords) do not usually even take part in the retrieval process at all. This reduces retrieval time, but

also allows for better ordering of the retrieval results by discarding irrelevant terms. Similar approach is used in general for other text mining tasks, such as classification.

Music Information Retrieval Evaluation eXchange (MIREX) has been created to compare and evaluate algorithms that operate on music data. It provides a TREC-like [137] collaboration framework for researchers dealing with both audio and symbolic music problems. Since the beginning, the Symbolic Melodic Similarity (SMS) task has its well-established position in a core of symbolic music analysis with repetitive releases among the years. We have used the released 2005 SMS MIREX dataset to evaluate how various design decisions impact performance of similarity measures, and we have contributed to the 2011 SMS task to evaluate further some aspects of using different term weighting approaches. The 2005 SMS MIREX task used a subset of a larger Répertoire International des Sources Musicales (RISM) music excerpts collection, introduced earlier as `rism` dataset, while the 2011 task was based on Essen Folksongs Collection (EFC) that contains more than 5,000 monophonic folk melodies (`essen` dataset).

3.6.1 Background

Previous Work

Existing approaches to symbolic music similarity task are common with the other tasks dealing with symbolic music. Among them one can differentiate several main streams, which will be described in the following paragraphs. Successful approaches can be found in all these groups, but string methods that deal with music in the similar way as text IR with written text seem to play a major role.

Geometric methods usually treat music excerpts as points in two-dimensional time and pitch space [101, 108, 109, 188, 190]. Since there is usually no notion of succession of notes (it is represented naturally on the time dimension) those methods are usually suitable for both monophonic and polyphonic music. However, it is generally more challenging to implement methods that are robust with regards to small changes that

are perceptually not very important, but cause the entire geometric representation to bend or stretch significantly.

String methods deal with music data as it is a linear sequence of symbols and use methods originally developed for texts. They do not typically handle polyphonic music very well, but SMS MIREX task deals with monophonic queries to the monophonic database, so this simplifies the problem for those methods. Among them, the major group employs algorithms based on the local or global alignment techniques, created initially to compute the edit distance between two strings [48, 49, 61, 69, 108, 176, 195, 196, 201, 202]. This seems to be a natural choice for the kind of data present in EFC and RISM datasets, since they contain only short melodies. The major drawback of those methods is their computational complexity, being at least $O(n^2)$. The other approach utilizes a “bag-of-words” type of analysis, that treats features as a set, without specific order between them [140, 175, 176]. This approach works best for larger documents giving the capability of linear processing ($O(n)$), which means that RISM and EFC datasets with their short excerpts might not be the perfect environment to show their benefits. Computational complexity for each query can be even sub-linear if we allow for initial indexing of n-gram features, bringing them very close to traditional text information retrieval systems.

There are also other ways to compute music similarity based on graph methods [154] or tree representation [161], but they are not as popular as the previous approaches.

Datasets

We have decided to base our work on two datasets that were previously used in the SMS MIREX tasks. The dataset from 2005 is based on a subset of Répertoire International des Sources Musicales (RISM) collection and contains two subsets used for training and evaluation, each having 11 unique queries and more than 550 melodies to match. The ordering of the most relevant results annotated by experts for each query has been analyzed by Typke et al. [186] and further refined and validated by

Urbano et al. [203]. This gives a solid background, allowing for complete evaluation of similarity algorithms, and the fact that the dataset have been released to the broad public make it suitable for performing our own analysis. The incipits in RISM collection are typically 10 – 40 notes in length.

The second dataset, Essen Folksongs Collection (EFC) has been used in MIREX SMS tasks in 2007, 2010 and 2011 edition. It contains 5,274 MIDI encoded monophonic folksongs from different countries. Typically the documents in the collection are longer than in RISM dataset, ranging from 15 to 80 notes with the average of 48 notes. For EFC based tasks, MIREX does not post any detailed evaluation data, including queries that are used to evaluate submitted algorithms (6 base queries, in 30 variants in total), and only quantitative performance data (rankings and performance measures) are publicly available. This allows for reuse of the same testing dataset among different releases of the SMS challenge, since the effort of creation of such dataset is significant as it requires lots of experts time. We use results from this dataset to verify hypotheses drawn from RISM dataset.

MIREX SMS Evaluation Framework

The evaluation of an SMS MIREX task takes place on ISMIR Conference. Teams submit their algorithms and each team is allowed to submit many variants of their submission. Each system is supposed to return the 10 most similar documents for each query, ordered by similarity to the query. Results are then combined and presented to human judges. Each time a judge sees a query melody and a single retrieved result (a pair of documents), they evaluate similarity between them. The system ensures that each pair is evaluated by a random grader, and receives from them two scores, ordinal (Very Similar, Somewhat Similar, Not Similar) and fine (a number from 0 to 100) indicating how close are those two melodies. The results lists are then evaluated using several performance measures, including Average Dynamic Recall (ADR), Normalized Recall at Group Boundaries (NRGB), Average Precision (AP),

Precision at N Documents (PND) plus a variety of summative measures that aggregate judge ratings for each returned result.

In our testing we focus on two measures. First of them is Average Dynamic Recall (ADR). To calculate it for each query, a union of all results returned by all candidate algorithms is obtained. The designers of SMS task assumed that if a few documents have very similar relevance ratings to a given query, their order in the perfect result list should not matter, e.g., a result list (equivalence groups indicated by brackets) $[ABC]D[EF]$ should yield the same result as $[BCA]D[FE]$. To calculate ADR, recall at each item on the result list is calculated ($r_i, 1 \leq i \leq 10$) assuming the results in the same group could occur in a different order, and ADR is an average of all r_i . In brief, ADR gives the average recall among all the documents that the user should have seen at any number of retrieved items [187]. It captures both quality (relevance of items) and order (how high are the most relevant items) aspects of the resulting ranked lists and it is a primary measure used in the evaluation of submitted algorithms to MIREX SMS task. For details on ADR measure please refer to [187].

The second measure we will evaluate the algorithms against is Fine Precision at 10 (FP10). It is the sum of all the fine ratings of all the items in the result set. It does not tell anything about the order of the results, but it indicates the general quality of all the results returned by the algorithm. Unlike for other measures, MIREX publishes not only total cumulative measures for all the queries, but also per-query FP10, allowing for analysis of differences in performance for different queries.

3.6.2 Methodology

Our approach to symbolic music retrieval focuses on evaluation of the impact on performance of certain aspects like feature extraction, n-gram length and similarity function. The basic framework of the retrieval process is kept standard. Our goal is to check what kind of results one can obtain with pure, well established methods known from text information retrieval ported directly to music data.

The main analysis has been conducted using the MIREX 2005 dataset, which consists of a subset of RISM collection. Based on that and using average dynamic recall (ADR) as a performance measure, we were able to evaluate the best parameters for each of the proposed approaches and draw general conclusions about the behaviour of the systems under different settings.

We have initially assumed that the performance may depend on the following factors:

- the feature extraction method used to retrieve basic components of the melodies in question (the analysis checks for importance of rhythmic and melodic components separately),
- the n-gram size, i.e., the length of the sliding window used for extraction of terms,
- the similarity measure applied to calculate similarity between term vectors representing two compared melodies, and
- the term weighting algorithm.

Our hypothesis is that all of those components play a role in retrieval of similar melodies, and our goal was to investigate the importance of each of them. From the plain melodic intervals and rational IOR, there are multiple approaches one can generalize the data or smooth out irrelevant differences, which we have indicated above. Without experiments, it is not possible to foresee what does and what does not influence the quality of the end result. Therefore our set of experiments covers various aspects, from quantization of rhythm and melody to the actual similarity formula.

The process of document retrieval starts with extracting features from documents. Input dataset consists of a set of standard MIDI files, each containing a single track of notes representing a monophonic melody. Since none of the notes are concurrent or overlapping in the datasets we have used, string-based methods can be directly applied to the input documents. Like text documents, that can just be seen as series of

characters, monophonic music pieces are just series of notes. The difference with music files is that text documents are easily separable into basic terms — words. Since there is no such a thing as a clear phrase boundary in music, the typical approach to bag-of-words, (or bag-of-terms) methods consists of building n -grams, i.e., all substrings of n consecutive tokens (notes). This process is widely used in bio-informatics (DNA sequence analysis) and in some text processing tasks (for authorship attribution [93], or for tasks with languages that have no word boundaries, like Thai [73]).

Granularity of Melodic and Rhythmic Features

Each of the basic features that conforms an n -gram (which is just an n -gram with a length of 1, or a uni-gram), is derived from each note event from the notes stream. It can either contain values representing absolute features, such as note's pitch, duration or inter-onset interval (IOI), but it is more beneficial to use relative features. They allow to achieve basic transposition and tempo invariance, which is required in this task, i.e., a melody played in a different key and in a different tempo, then the base melody, should be considered equivalent. Therefore we have decided to use melodic intervals and inter-onset interval ratios (IORs). The other question is how one should translate the numbers that represent melodic intervals and IORs into discrete values, i.e., at which level of granularity they should be dealt with.

Melodic intervals derived from MIDI files give precise, discrete interval classes. We have chosen the following levels of granularity of melodic features to test:

- accurate/fine: each feature represents the actual interval between two consecutive notes, in semitones, taken directly from MIDI note events.
- coarse: intervals belonging to the same class are grouped together. We use five classes: same (no pitch change), small jump up (1-3 semitones), small jump down, large jump up (more than 3 semitones), large jump down.
- contour: only direction of the melody matters. This gives three classes: no change, melody ascends, melody descends.

- identity: always 0. Melodic properties are excluded from feature extraction

Raw inter-onset interval ratios (IORs), unlike melodic intervals, represent rational numbers. The method used in this submission uses our previous research results where rounding (with 0.2 threshold) was applied to the binary logarithm of the IOR. This gives progressively wider steps as IOR increases, yet being precise enough to maintain the perception of rhythm changes [216]. With this as a base, we have come up with four IOR granulation schemes:

- accurate/fine: rounded values of binary logarithm of IORs with precision of 0.2.
- coarse: five classes of no change in duration ($\log_2 IOR = 0$), next note being at least twice as fast ($-1 \leq \log_2 IOR < 0$), at least twice as slow ($0 < \log_2 IOR \leq 1$), more than twice as fast ($\log_2 IOR < -1$) and more than twice as slow ($\log_2 IOR > 1$) as the previous note.
- contour: similarly to melodic contour, with three classes — same duration, slower or faster.
- identity: always 0, which excludes rhythmic properties.

Length of n-grams

With our testing we were also able to determine the optimal n for each of the proposed settings. We have analyzed n-gram lengths from 1 to 10 with peaks of performance observed usually between $n = 2$ and $n = 7$. The general rule of thumb is though, the more general the features, the bigger the n should be.

Similarity Measure

We have also tested for the impact of the following similarity measures:

- Common Features: represents a dot product between two vectors representing two documents that are being measured, which can be denoted as follows:

$$sim(\mathbf{x}, \mathbf{y}) = \sum_i x_i y_i \quad (3.16)$$

where x_i is a weight of a term i in a document \mathbf{x} .

- Cosine Similarity: the cosine of the angle between both document in the high-dimensional feature space. In essence it is a dot product between two vectors normalized by their lengths. Unlike the previous approach, this also takes the length of the vectors into consideration:

$$sim(\mathbf{x}, \mathbf{y}) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2}} \quad (3.17)$$

- CNG measure: This method calculates the arithmetic mean over n -gram vectors that consist of weighted features. The similarity measure is proven to distinguish well between authors of texts [93] as well as composers in music domain [216] making it an interesting candidate for application to this task. The equivalent formula for the similarity measure is given as follows:

$$sim(\mathbf{x}, \mathbf{y}) = \sum_i \left(1 - \left(\frac{x_i - y_i}{x_i + y_i} \right)^2 \right) \quad (3.18)$$

Term Weighting Method

One of the goals of this research was to measure impact of how various text-based term weighting measures affect measuring similarity between music documents. We have decided to evaluate four approaches:

- binary: It is either 0 (if a term, or an n -gram does not appear in the document) or 1 (if a term appears in the document). With Common Features similarity measure, it gives a basic number indicating the number of terms two documents have in common.

- frequency: simple term counts (the number of times the term appears in the compared documents) which gives a classical cosine similarity or CNG measure definition.
- tf.idf: a standard term weighting technique where term frequencies are normalized by document frequencies, i.e., the number of documents a term occurs in. This penalizes high frequency, common terms that occur in most documents. The formula for an term i in the document x is given as follows:

$$tf.idf_{x_i} = \frac{count_{x_i}}{\|x\|} \log \frac{\|D\|}{\delta_i} \quad (3.19)$$

where $\delta_i = \|\{d \in D | i \in d\}\|$ is the number of documents containing term i in the entire collection D . This measure is commonly used in Textual Information Retrieval for term weighting so it would be interesting to see how it performs for music data.

- Okapi BM25: it is an industry-developed weighting scheme, that typically outperforms classic term weighting measures like tf.idf. It tries to capture roughly the same concept as original tf.idf measure but attempts to balance documents with different lengths and different term distribution:

$$bm25_{x_i} = \frac{count_{x_i}(k+1)}{count_{x_i} + k(1-b + b \frac{\|D\|}{avgdl})} \log \frac{\|D\| - \delta_i + 0.5}{\|D\| + \delta_i} \quad (3.20)$$

where $avgdl$ is an average document length. It is parametrized, with parameters b and k , and we have used recommended settings of $b = 0.75$ and $k = 2$.

3.6.3 Analysis Based on RISM Data

In order to evaluate which factors have the biggest impact on the performance of similarity algorithms, we have reproduced the MIREX 2005 task using existing document relevance information. At each run of an algorithm, a set of 10 ordered results is returned. For each of the results a relevance score is assumed, such that if a result

was previously marked as relevant with an appropriate relevance group label used to calculate ADR score, the same group is kept. If a result relevancy to a given query was not marked in the judgment list, the item is assumed to be not relevant, although it could get some recognition from the judges if it was submitted to the actual MIREX task. As a result of that, the measured ADR score does not exceed the actual ADR score that the given result set would get, so it marks its lower bound.

For each similarity measure test, the ADR score is evaluated for each available query and the average ADR among all queries is returned as a result. The performance for each different settings is tested for a range of n-gram length values, although one could use one of the other dimensions as a reference point (e.g., feature granulation) as well; it is an arbitrary choice.

We have found that, regardless of the term weighting algorithm applied, melodic features give better results than rhythmic features and that there is not a significant difference in using only melodic features versus combined melodic and rhythmic (see Figure 3.15). One can observe that the peak performance is achieved with n-gram lengths from $n = 2$ to 3 for more precise, combined features, and between $n = 3$ to 5 for more general melodic features. Using only IOR's (rhythmic) features leads to significantly worse results with the peak performance around $n = 5$ or 6.

The significance of the choice of feature granulation scheme is shown on Figure 3.16. The conclusion is that the finer (or the more precise) the representation, the better results can be achieved with smaller n . Figure 3.16 shows the performance for melodic features with increasing generalization of intervals. The peak performance is achieved with fine interval values for $n = 2$ and 3. Coarse representation, with only five levels yields the best results around $n = 5$ and melodic contour (three levels) performs much worse, with a peak performance around $n = 7$.

Figure 3.17 shows ADR scores achieved using fine melodic interval features using different similarity measures (CNG, common features and cosine) and Figure 3.18 shows scores for cosine similarity measure and with different weighting methods applied (binary, frequency, tf.idf and bm25). It turned out, rather surprisingly, that the

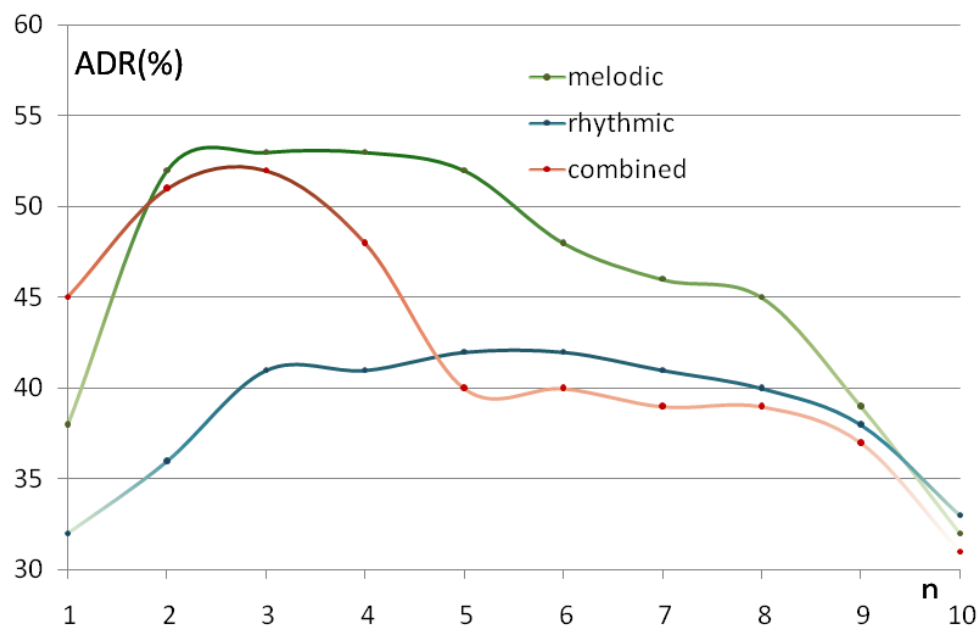


Figure 3.15: The comparison of performance of similarity measures using the same cosine similarity method, and using various feature extraction method. Using only rhythmic features is easily outperformed by melodic features and combined melody with rhythm.

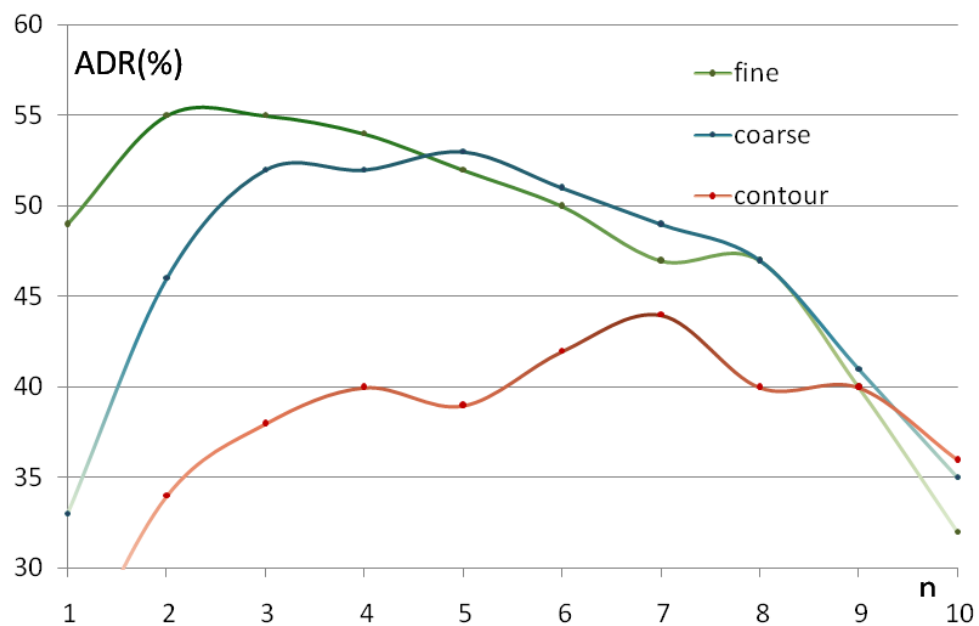


Figure 3.16: The influence of various quantization approaches for the same feature extraction method (here, melodic intervals). More precise features offer better performance than more general ones.

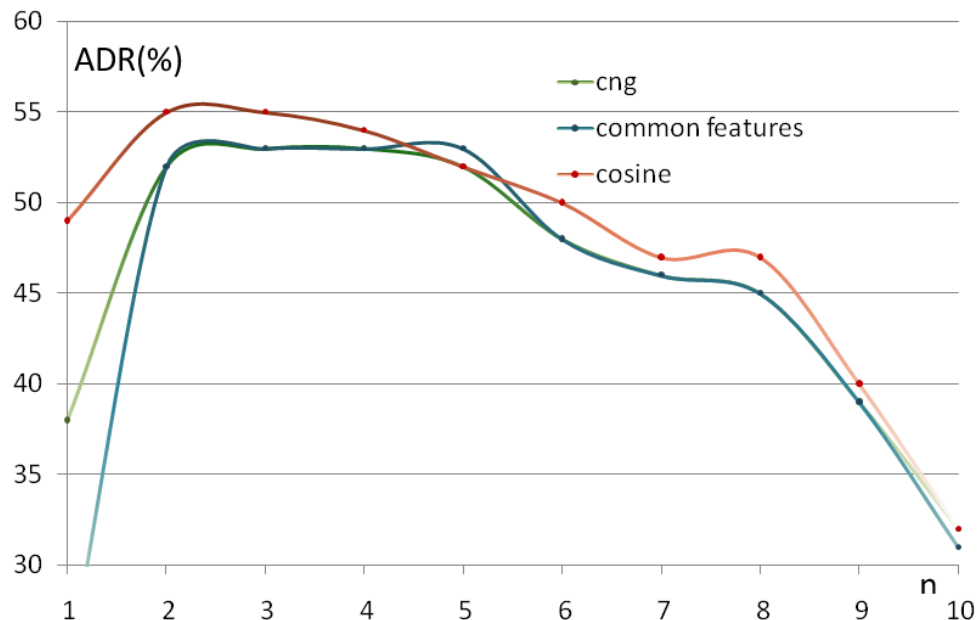


Figure 3.17: Here the same fine melodic interval features have been used with three different similarity measures. As one can see, there is no significant difference in performance with regards to the similarity function applied to measure similarity between music excerpts

usually important aspects, do not have much influence on the final result, e.g., the number of features in common gives as good results as applying cosine similarity measure with bm25 term weighting method. The peak performance for all those methods is achieved between $n = 2$ and 4. This can result from the fact, that queries and documents are rather short and those more sophisticated similarity measures were designed to evaluate similarity between larger documents or profiles with hundreds and thousands of features.

The MIREX 2011 Evaluation

Having in mind our previous findings, we have chosen to evaluate further how different text-based term weighting methods perform in a different task. To do that we came up with the following 6 setups, with parameters tuned according to our previous analysis. All of our submissions feature cosine similarity measure with melodic or

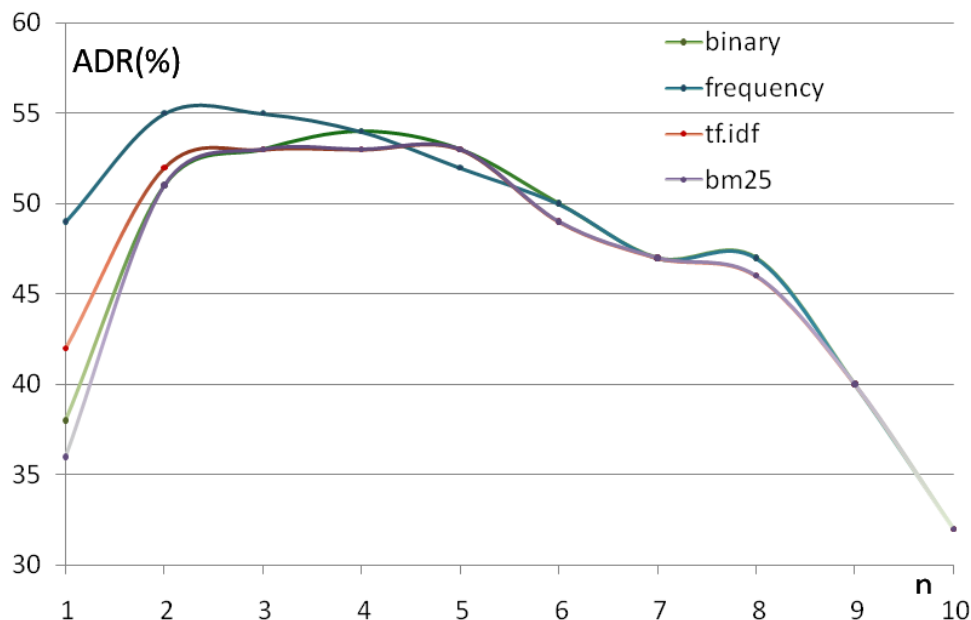


Figure 3.18: Fine melodic interval features have been used with cosine using different weighting scheme. Again, no significant difference in performance with regards to the weighting scheme applied to measure similarity between music excerpts

combined, fine features. The main aspect that varies between them is the term weighting approach:

- Binary (WK1).** It uses binary term weighting approach and fine melodic features with $n = 5$. The resulting similarity measure between two documents is the number of terms in common normalized by geometric average of numbers of unique terms in both documents. This algorithm got the best results on the 2005 SMS MIREX dataset.
- Term counts (WK2 and WK3).** They both use frequency-based weighting, which gives a classical cosine similarity definition. This rather simple method gave us surprisingly good results for two different settings so we have decided to submit both for MIREX evaluation. The first one (WK2) uses again melodic intervals as features and $n = 4$, while the second one uses a combinations of melodic intervals and IORs with $n = 2$. The relative performance of these two

algorithms allow us to assess whether introduction of rhythmic features helps to improve the overall score.

- **tf.idf (WK4)**. The algorithm computes standard tf.idf weights of each term from documents to compare, which gives each term weight depending on its count within the document and in how many documents of the collection a term occurs. For the settings of WK4 we have determined, that $n = 4$ and melodic interval features worked the best in our tests.
- **Okapi BM25 (WK5 and WK6)**. Since they feature bm25 weighting, and it may be a top performing function, we have came up with two sets of settings: WK5 with melodic interval features of length 4 and WK6 with features combining melodic interval and IORs with n-gram length of 2.

Our algorithms were evaluated along with 5 other submitted algorithms reaching similar total score. UL set of algorithms uses sequence alignment with geometric representations (see Urbano et al. [202] for details), while LJY algorithms employ quite similar approach with coded geometric melodic contour model (see Lee et al. [106] for details).

Only the UL series algorithms outperformed most of our submissions, yet still the difference in most cases was not measured as significant (apart from UL1) [100]. For most measures, only cumulative results were published, which does not allow us to draw many conclusions about the actual algorithm performance, however for the purpose of performing Friedman test with multiple comparison results, a FP10 results for each query and each algorithm have been published. FP10 stands for fine precision at 10 and is the sum of all the fine ratings of all the items in each of the result sets. The results for each query type are collected in the Table 3.3 (ADR measure) and Table 3.4 (FP10 measure). The best and the worst performers for each query have been highlighted (with bolded and crossed-out cells respectively) and all the values — colour coded for clarity.

According to the results, with ADR scoring, UL3 was the best in all 5 categories and was the best algorithm for this measure overall. UL1 and one of our submissions,

Table 3.3: Results of SMS 2011 task calculated for each query modification type separately. The numbers represent average dynamic recall (ADR) values, in percentage. Other participating algorithms are described in Urbano et al. [202] and Lee et al. [106].

	LJY1	LJY2	UL2	UL1	UL3	WK1	WK2	WK3	WK4	WK5	WK6
overall	64	66	65	68	73	68	67	66	65	65	65
no errors	67	69	68	70	72	68	69	67	68	69	67
deleted	67	68	62	69	76	70	67	65	64	65	62
inserted	59	61	66	66	70	65	64	64	63	62	65
enlarged	64	65	64	69	73	67	66	65	63	64	65
compressed	65	67	65	63	72	67	67	67	67	67	67

Table 3.4: Results of SMS 2011 task calculated for each query modification type separately. The numbers represent fine precision at 10 (FP10) values, in percents.

	LJY1	LJY2	UL2	UL1	UL3	WK1	WK2	WK3	WK4	WK5	WK6
overall	48	49	57	59	55	51	49	43	46	47	46
no errors	50	52	60	63	56	55	51	45	50	49	49
deleted	47	50	60	63	56	52	47	43	42	44	48
inserted	49	48	56	60	56	50	50	42	46	47	46
enlarged	47	47	54	54	56	47	48	42	45	43	40
compressed	48	49	55	57	52	53	48	44	48	50	46

Table 3.5: Results of SMS 2011 task calculated for each base query separately. The numbers represent FP10 values, in percents.

	LJY1	LJY2	UL2	UL1	UL3	WK1	WK2	WK3	WK4	WK5	WK6
q01	42	48	62	64	44	47	50	44	48	47	52
q02	69	69	73	70	71	72	56	42	44	44	24
q03	43	51	56	62	50	68	62	31	56	62	46
q04	50	48	56	57	61	56	41	53	48	44	61
q05	45	46	56	64	64	52	56	48	53	56	70
q06	49	49	58	57	45	36	41	53	52	40	39

WK1 came second, on par. In terms of precision (FP10, Table 3.4), all UL algorithms outperformed the competition by at least a few percents.

What drew our attention were the fine results calculated for each query separately (see Table 3.5). The table consists of FP10 values achieved by each of the algorithms for every base query which in essence breaks down the *no errors* row from Table 3.4. It turned out that a lot depends on the actual query, since our most sophisticated setup — WK6, although it performed rather poorly overall (it was one of the algorithms that came last in this category), achieved the best scores in two out of six queries. Since one knows nothing about the actual queries (this also is kept confidential at MIREX) it does not allow us to draw any meaningful conclusion why it happened, but one can clearly see that the type of the actual query should also play an important role in determining the best algorithm for the task.

3.6.4 Conclusions

The results of our experiments show that for this simple tasks even basic retrieval methods yield very satisfactory results. At the end, the simplest of our algorithms submitted to 2011 MIREX SMS challenge came second overall. On the other hand, we have pointed out the importance of more foundational design decisions like the use of proper features extraction method, and the impact of choosing a proper n-gram length. The FP10 per query results from MIREX 2011 show that there are other

factors, that we are just unable to capture because of lack of proper evaluation data, which creates more possibilities for the future research.

3.7 Summary of the Chapter

In this chapter we have shown our n-gram-based approach to symbolic music. It deals with music in a similar way as with text and allows application of similar methods to common tasks, like classification or search. We introduced and collected a number of classical music corpora, which contain symbolic music data. Since music is a complex type of data, it requires additional treatment upon pre-processing so we have presented a PERL module, `MIDI::Corpus`, that implements essential data handling, feature extraction and analysis functions required in many, typical data mining tasks, also introduced in this section.

With the data and the tools, we were able to analyze various statistical features of symbolic music in comparison with several text corpora. We found a great resemblance of symbolic music and text, which justifies application of text-derived methods to analysis of symbolic music. This led to the analysis of the relevance of several text-based techniques on tasks vital for both, text and music domain: classification (authorship attribution) and retrieval (measuring document similarity).

For the first task, we analyzed several similarity metrics with leave-one-out cross-validation on three corpora and compared the results to human judgments as well as results obtained by others. We have shown, that with proper parameters, text-based measures were capable of obtaining very good results, outperforming human scores and being on par with other automatic methods.

In the second task, we implemented a n-gram-based retrieval system, being able to find melodies similar to a given query in a large corpus. The system has been submitted to the 2011 MIREX symbolic melodic similarity task, achieving good results, marginally worse than the top performing algorithms, while still allowing for great scalability and possible indexing for sub-linear performance. The detailed analysis of the factors determining the optimal performance revealed that it is more important

how basic terms are constructed from raw notes data (such as the choice of n-gram length, representation of music features), than the actual selection of similarity measures for retrieval.

Having shown, how text-based methodology functions in typical text mining setting, the focus in next chapters is shifted towards more music-specific applications, as melody generation, visualization and structural analysis, which would allow to show, how robust is our methodology for a different, not text-related kinds of problems.

Chapter 4

Using Genetic Algorithms for Evolving Excerpts with Corpus-Based Fitness Evaluation

This chapter addresses the issue of automatic generation of music excerpts. The character of the problem makes it suitable for various kinds of evolutionary computation algorithms. We introduce a special method of indirect musical melodies representation that allows simple application of standard search operators like crossover and mutation with no repair mechanisms needed. We propose a method for automatic evaluation of melodies based upon a corpus of manually coded examples, such as classical music pieces. Various kinds of Genetic Algorithm (GA) systems were evaluated i.e., generational GAs, steady-state GAs and multi-objective optimizations. The problem of determining fitness landscape for this problem is also addressed. The results identifies the potential and the issues associated with certain approaches for further applications in the domain of automatic music composition.

4.1 Background

Internally, music is a well-structured organization of notes. Thus one can design algorithms and systems to make computers understand music and aid humans in their composition tasks. Applications for music touch many fields of computer science, but evolutionary computation seems to be especially suited for music; not least because music melodies have a linear, sequential nature, which clearly lends itself to representation in terms of a gene structure. Moving beyond representational issues, there are still many outstanding problems that are worthy of study. A few major problems to address were summarized by McCormack in [126] and this chapter propose a solution to at least two of them: the problem of building music representation

especially suited for evolutionary techniques and development of automated fitness function that is able to select “pleasant” individuals.

Automatic evaluation of melodies in terms of their fitness is both the most important and the most difficult problem. A judgement whether certain melody sounds ‘good’ or ‘bad’ is a vague issue even for humans. Many solutions to this problem try to overcome it by manual fitness assignment, but this compromises our ability to automate the process of music creation. Other approaches include fitness-assigning methods that result from some theoretical models of music. However, aside from the fact that such models are very hard to derive, limiting melody creation in such a way results in a series of artificial constraints on the resulting composition. The third approach, which will be used in this chapter, utilizes sample pieces from users to create classifiers able to judge the properties inherent in new melodies.

There are many other problems such as representation issues or application of different search operators that must be solved. However, assuming some basic music theory, those problems can be easily addressed, yet still requiring special, music-designated solutions. This determines the uniqueness of music-driven tasks. And it means no “free lunch” for those who try to solve music related problems using standard methods and approaches.

4.1.1 Previous Work

Since the structure of music is linear and quite easy to represent in Genetic Algorithms (GA), several works have considered solving music generation problems using evolutionary methods.

One of these approaches was introduced by Biles [12], who presented a system for generating jazz solos. It is based on developing two populations (phrases and bars) of music melodies. The authors used a fixed structure of the music and fixed and small set of possible notes which resulted in significant simplification of the problem. They relied on human judgement as a feedback for fitness evaluation, which limits the number of iterations and population size in GA and thus the application and

thorough testing of the system. As the problem of generating jazz solos becomes popular, there are approaches that try to address the human factor problem [3].

Jacob [87] proposed a complete system for composing music. This system consists of several GA agents, each responsible for a different stage of the work of a composer (e.g., phrase selection, structure arrangement). However, this system also relies on human judgements and it also requires a set of initial themes as building blocks for the system. As a result, the author presented a set of compositions made by the system. However, the structure of the agents and the way the agents work were not described in the paper. Similar approach was proposed also by Gartland-Jones [58].

Johanson and Poli [92] proposed a method for using Genetic Programming (GP) for generating small melodic sequences. This approach also required manual user evaluation of every generation, but they also created a neural network that used the human feedback from their previous experiments for automatic evaluation of later melodies. The neural network was able to provide as good an assessment as the human evaluation. A similar approach was presented by Tokui and Iba [180] for generating rhythm structures using both GA and GP with human judgement-based fitness evaluation. They reported good results, which could result from several factors: the rhythmic domain is much simpler than melodic one (as indicated in the previous chapter), they used complex architecture to solve the problem, and they employed human feedback to assess fitness — accurate but slow, expensive and not a scalable solution.

Papadopoulos and Wiggins [143] pointed out that there is no proven approach for establishing the automatic evaluation of music quality. They proposed a method of evading this problem by applying automated fitness evaluation that was based on static melody features. The automated fitness evaluation based on several static melody features (such as contour and speed) provides specific melodies with a nice ‘overall look’ without reference to the human perception of a piece. A weighted fitness of all the melody features was assumed. However, a Pareto-based evaluation, which

takes all aspects of the system as equally important and evaluate them on separate dimensions, might be more appropriate in this situation.

An approach similar to the one used in this chapter can be found in Manaris et al. [122,123] and Machado et al. [116]. They also use a corpus of recognized “pleasant” music as a foundation for development of a fitness function to assess individuals in a GA process. They use corpora to create certain ideal metrics based on Zipf’s statistics of various music features, e.g., intervals or IORs, typically without combining them into longer, n-gram sequences (tri-grams were the longest observed sequences). Despite their aims to build a definite fitness function for genetic systems, they applied their techniques primarily to classification tasks. Our goal is to build a model based on a corpus of existing music pieces directly for evolving music excerpts through a GA process.

Other approaches to the problem of automatic fitness assignment for melodies generation involve Neural Networks [139], SOMs [103, 151], fixed musicological rules [94,217] or similarity to a target [58]. There are also approaches to polyphonic music generation [59, 123, 181], not studied in this chapter.

4.2 Methodology

Evolutionary computation bases upon premises of genetics, evolution and natural selection known from Charles Darwin’s evolutionary biology. The basic building block of the system is an individual represented by a genome — a series of features (genes), like parameter values, steps or instructions, indicating a complete solution to a given computational problem. A certain number of those individuals form a generation. From each generation, pairs of individuals are chosen based on their fitness to mate (exchange their gene material) to generate offspring (new individuals) that can further mutate (randomly change their genes, typically at a very small rate). Entire offspring from a certain generation forms the next generation and the process repeats until a specified goal is met, that is, we obtain individuals that satisfy our predefined quality requirements for the final solution.

This chapter presents a novel approach to representing melodies (individuals in GA terms), and avoids utilizing human feedback during fitness evaluation. However, a sample of MIDI files is necessary to provide the source from which music features are extracted. Specifically, a complete set of Preludes and Fugues from *Das Wohltemperierte Klavier 1* by J.S. Bach establishes the source of music features, although no special significance is associated with this selection.

Similarly to other work done in this area, the melodies created by the system operate on the Western music twelve-tone scale. However, unlike previous approaches, where notes were represented using absolute note pitches and timings [58,94], here the smallest melody component will be the information about pitch change and duration change. The genome would be a sequence of genes, dubbed unigrams — pairs of integer values indicating the interval between notes in semitones (i.e., melodic intervals) and the rounded binary logarithm of the ratio of the corresponding note's duration (i.e., inter-onset interval ratios (IORs) in MIR nomenclature).

Naturally, the same melody can be played in different scales or different tempos and it still remains the same melody and preserves relations between notes. By emphasizing the underlying relation between notes we fulfill the human perception of rhythm and melody. Since the pitch and the tempo of a melody is not determined nor needed, the preset scale and the tempo are used in playback only.

The other advantage of this approach over direct representations is that the logic of the melody is preserved. Moreover, application of classical one/two point crossover does not result in lethals, if one creates interval-based children. Note-based children, which result from using direct representations, may create odd connections at crossover points, like the 9th on Figure 4.1, that does not exist in any of the parents. This typically makes the resulting melody corrupted, and thus the individual — lethal.

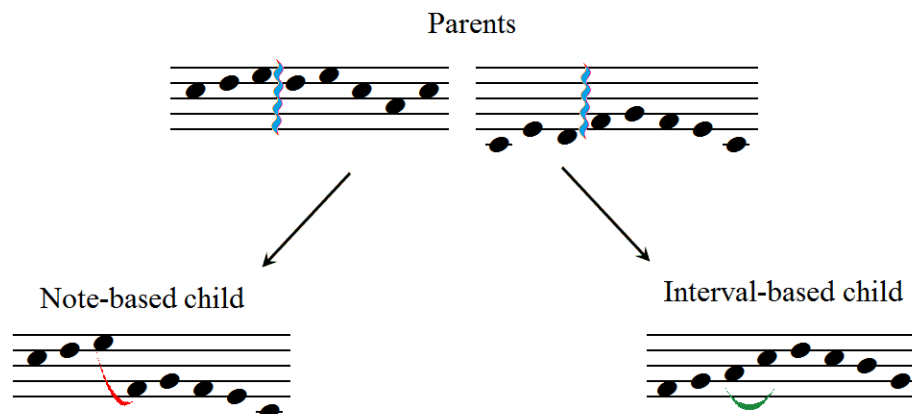


Figure 4.1: Two children created using different representations for standard crossover.

4.2.1 GA Design Decisions

The evaluation of fitness is automatic and reflects the likelihood that the given melody is a good melody according to a given corpus of musical compositions. The features of the corpus are calculated by counting all different sequences of melodies of n consecutive unigrams, i.e., n -grams, and taking some statistical information of n -grams in the corpus like:

1. n -gram's tf (term frequency, n -gram's count)
2. n -gram's probability (normalized count)
3. n -gram's df (document frequency, the count of documents in which the n -gram occurs)

Based on the statistical information from the corpus, fitness can be evaluated according to the following general formula:

$$Fitness(\text{melody}) = \Gamma_1 \left(w_n \cdot \Gamma_2 \varphi(n\text{-gram}) \right) \quad (4.1)$$

where:

- Γ_1 and Γ_2 are some aggregate functions e.g., sum, average, max;
- Γ_1 has the range over the various n -gram sizes, and

- Γ_2 has the range of all n-grams in an individual with the size of n ,
- w_n is a weighting factor of certain n-gram size,
- $\varphi(\text{n-gram})$ is an n-gram weighting function that is derived from corpus statistics.

The formula is left in the basic form because various types of GA's presented in this chapter will use different component functions Γ_1 , Γ_2 and φ .

Selection operator. A standard roulette-wheel selection operator will be considered, where the likelihood of being chosen for reproduction is proportional to the fitness value of the individual.

Search operators. The indirect representation proposed in this system allows several classical operators to be supported directly:

- **Unbounded crossover.** One-point crossover is defined such that the crossover point is selected independently in each individual. That is to say, a melody coded using our proposed indirect method does not have the limitation where a gene at a position has a certain meaning according to its localization in the genome. Moreover, we do not want to impose a priori limits on the evolution of melody length. This approach allows creation of individuals of any size, albeit potentially leading to the phenomenon known as 'code bloat' from genetic programming.
- **Gene-wise stochastic mutation.** Mutation consists of replacing a gene with a new tuple drawn according to the probability of unigrams (n-grams of the length 1) from the corpus. Since unigrams represent a change of melody, drawing a melody from the corpus probabilistically will change a single inflection point of the melody leaving the rest of the melody (relatively) unaffected.

4.3 Experiments

The following experiments were conducted to characterize various features of the solution presented to the problem of automatic music melody generation. In most of the experiments the following algorithm parameters were set:

1. Population size: 100
2. Individuals in the initial population are sequences of 10 unigrams drawn randomly according to the probabilities obtained from the corpus
3. Equal w_n weights are used for evaluating different n-grams of different sizes.
4. N (maximal size of n-grams considered): 6
5. Probability of crossover: 1.0, mutation: 0.0
6. Number of epochs/generations: 100

All experiments were written as Perl scripts using MIDI::Corpus module for analyzing and managing n-gram features of MIDI files and individuals in the populations.

4.3.1 N-gram Statistics

The key point in this system is to define the details of the fitness function. The simple and most obvious decision for aggregate functions Γ_1 and Γ_2 is using the sum or the average for Γ_1 , and the average for Γ_2 . One will then have ranked individuals according to the average fitness of their components. The main challenge is how to evaluate individual sub-sequences for their likelihood of being good or bad melodies. The simplest approach is to take the probability of n-grams in a reference corpus as an indicator of fitness. However, despite the fact that ‘good’ melodies are usually quite complicated, the majority of the corpus is built from very simple elements. The melodies evolved from φ functions based on the probability model resulted in passages of notes just going up, down or simple trills i.e., the most frequent n-grams.

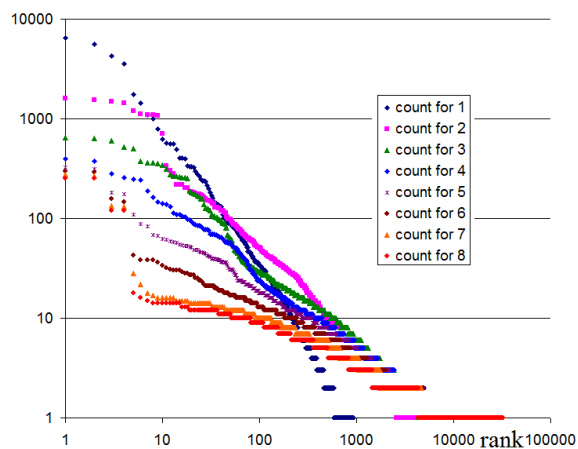
Such behaviour is similar to the occurrence of stopwords known from Information Retrieval (IR). The value of melodies resulting from using these kinds of n-grams is minor while the fitness reward for the melodies containing them is high. The distribution of the n-grams with a different size in the corpus follows Zipf's law as shown in Manaris et al. [122,123] as well as in chapter 3 of this dissertation. Figure 4.2(a) show this property on our reference Bach's WTK corpus. The most frequent n-grams are many times more popular than others. We have checked that replacing the φ frequency-based function with the binary existence test (i.e., 1 if an n-gram exists in the corpus and 0 otherwise) does not improve the situation so there should be a mechanism to remove the influence of the most common n-grams.

The simple, non-parametric and conceptually acceptable solution to this problem is to use a variant of the tf.idf measure from IR to determine term importance. Most important terms are the ones that are quite frequent, but occur in few documents. The most frequent n-grams are present almost in every document. Thus they are discarded as good melody components. Keeping in mind the exponential distribution of n-grams in a musical document, the following formula may be applied to express the 'goodness' of an n-gram:

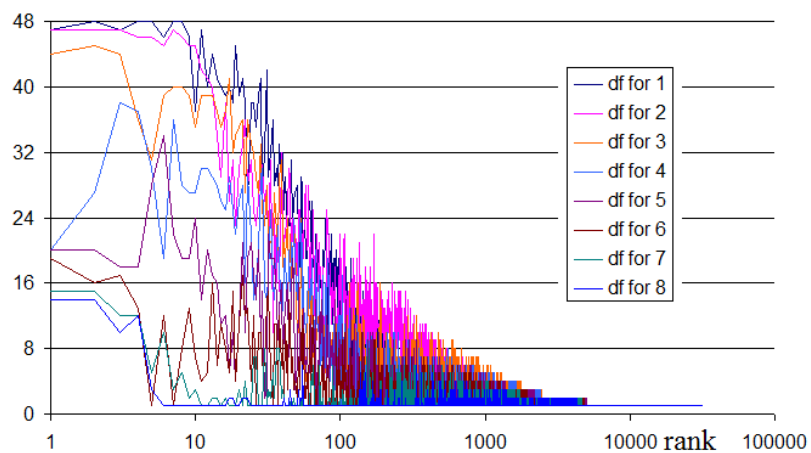
$$\varphi(\text{n-gram}) = \frac{\log(tf(\text{n-gram}))}{df(\text{n-gram})} \quad (4.2)$$

The distribution of tf.idf values according to n-gram's popularity is shown in Figure 4.2(c).

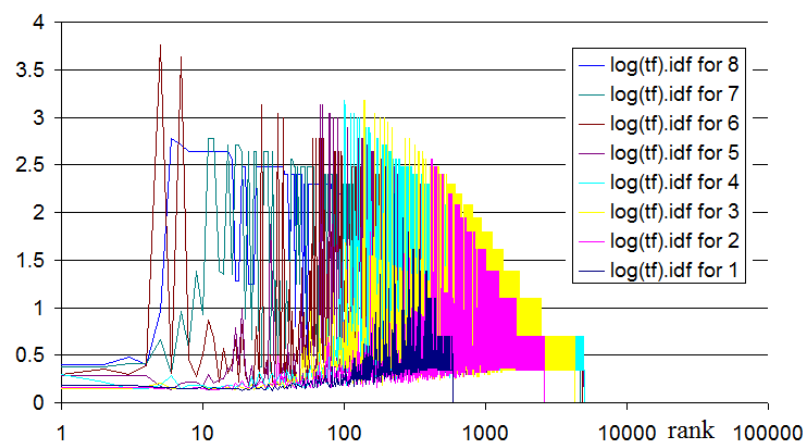
It is apparent that this approach successfully decreased the influence of 'stopwords' in the fitness function. The φ value of the most frequent n-grams was around 0.2 while for the most important n-grams this value reaches 4. However, the stopwords remained important because the smallest simple components may build larger, much fitter blocks. The influence of very low probable n-grams was also diminished, keeping the most valuable n-grams in the middle of the rank scale.



(a) Count distribution



(b) Document frequency distribution



(c) tf.idf distribution

Figure 4.2: Distribution of n-grams: counts (a), document frequencies (b) and tf.idf scores (c) according to their rank.

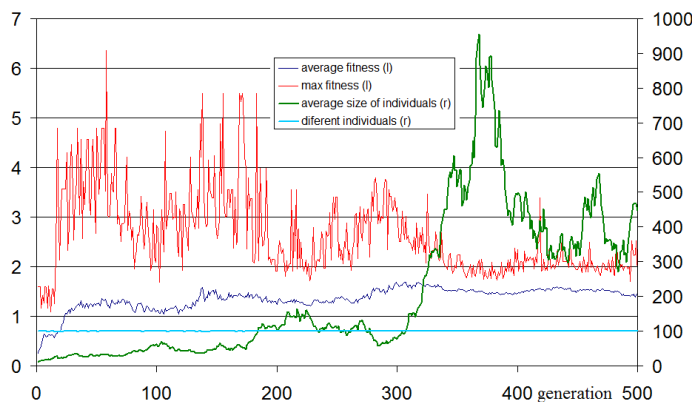


Figure 4.3: Generational GA evolving 100 melodies over 500 generations. Average fitness, maximum fitness, average size of melodies and the number of different individuals are plotted, with indications whether values for certain parameter are on left (l) or right (r) vertical axis.

4.3.2 Generational GA

Having established a suitably informative fitness function, we are in a position to apply a standard generational GA (GGA). Figure 4.3 shows a sample run of this system. Most parameters were set to default values. The parameters tracked were: average and maximum fitness, average individual’s length, and the number of different individuals in the population. The fitness function represented the sum (Γ_1) of five averages (Γ_2) associated with component n-grams (n from 2 to 6). The average values for fitness was around 1.5 which gives the average φ value of 0.3 that is close to the stopwords value. The maximum fitness was changing rapidly, reaching the level of 3 (average φ value of 0.6) without the ability of keeping these good genes across generations. It was still far from the peak φ values (~ 4.0). This showed the inability of the Generational GA to take advantage of tf.idf rating.

In later generations a new phenomenon occurs — ‘note bloat’, unlimited grow of individuals in the population, similar to code bloat known from Genetic Programming. It may result from many factors. The individuals are not limited in length. Crossover is known to have a destructive influence on the offspring. The influence of a wound after crossing over some ‘valuable’ regions may be hidden by the length of individuals (other n-grams). Hence shorter children are more likely to suffer from

this destructive influence than longer ones. The influence of note bloat is definitely negative. Populations with larger individuals typically register lower fitness properties (average and maximal). One can observe negative correlation of fitness and individual size.

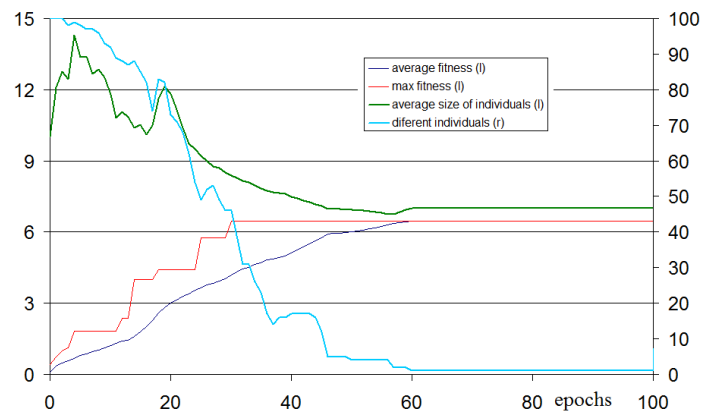
4.3.3 Steady-State GA

A steady-state, elitist selection operator always maintains the best solutions. Only one child is created per iteration and the worst solution from the current population is replaced. Since only one individual changes from generation to generation, to better reflect computational complexity of steady-state GA, epochs are used instead of generations. One epoch passes after a system generates the same number of new individuals as it existed in the original population. This gives the same execution cost as one generation of traditional Generational GA, and unlike for Generational GA, the fittest individuals can survive from one epoch to another, promoting elitism. An illustrative sample run of the system with this approach is shown in Figure 4.4(a).

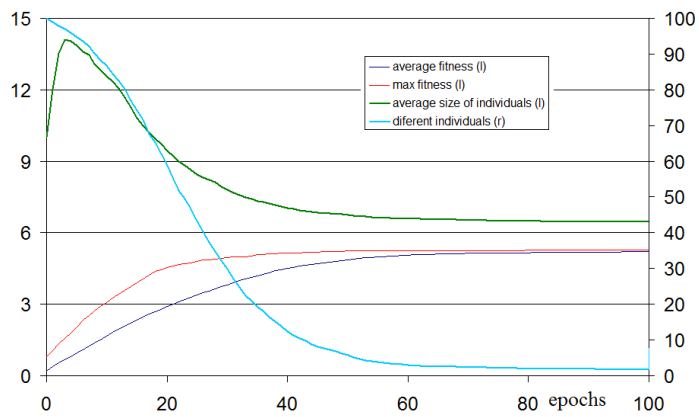
Unlike Generational GA, the variety in the population drops and the whole population ‘converges’ to a single individual. One can observe a quite quick convergence to one solution found around the 45th epoch. There is no note bloat and the quality of solutions is higher compared to those identified by Generational GA. Figure 4.4(b) shows the average behaviour over 100 runs. In the first 5-10 runs the algorithm searches for a niche to converge (increasing size of individuals); thereafter, a single solution appears.

Mutation

Mutation can inject new genes, resulting in new component melodies, which may be more or less fit than those currently in the population. Steady-state selection insulates us from the destructive influences of mutation. The averaged behaviour of the system with mutation is shown in Figure 4.5. Moreover, it appears that mutation is important; the rate of 0.2 results in a fitness improvement of around 20%, and with



(a) Single run



(b) Average

Figure 4.4: Steady-state GA evolving run over 100 epochs: single run example (a), and average over 100 runs (b).

even a small gene-wise mutation rate of 0.01 resulting in significant improvements. Without mutation, the steady-state approach concentrated only on a single peak in the landscape; resulting in premature convergence. The positive influence of mutation is due to the fact that with elitist selection, the solution tends to converge to a local optimum, and mutation helps to jump to a possibly better solution that is relatively close to the current best solution. However, applying mutation to Generational GA did not provide any difference to the results obtained by this method.

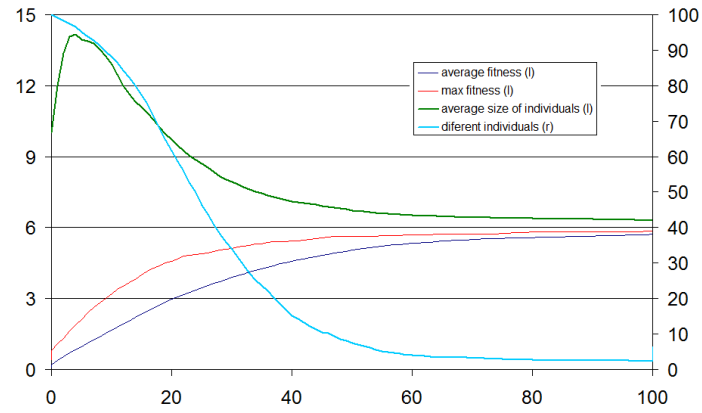
4.3.4 Diversity by Multi-Objective Optimization

The main problem of the simple single-objective steady-state approach is that it evolves to only one solution, which is usually not indicative of the potential spread of available solutions. One needs a method of preserving individuals that contain valuable n-grams, which may be the components of good melodies, but without assuming a single solution to the overall problem. The solution to this may be to utilize a multi-objective Pareto-based fitness function based on two criteria:

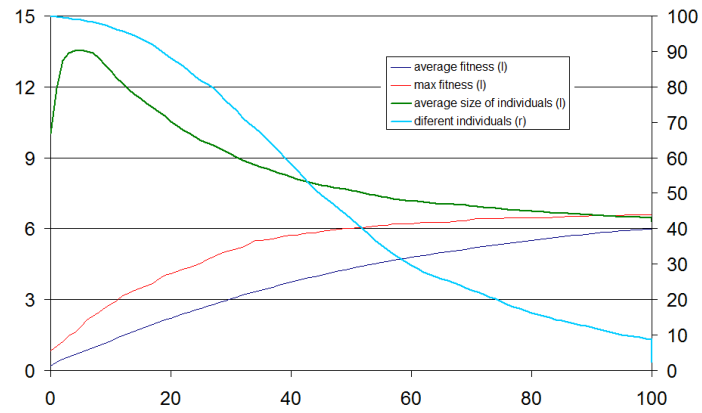
1. Standard fitness evaluated by the ‘average’ function as Γ_1 and Γ_2 .
2. Fitness evaluated taking the ‘max’ function as Γ_1 and Γ_2 .

The first objective will take care of the overall performance of melodies. The second will take care of those individuals that contain valuable fragments, but do not have enough other content to record a high average value. Those individuals may be crossed over with those with a high standard fitness; thus establishing a chance of injecting their material to the offspring.

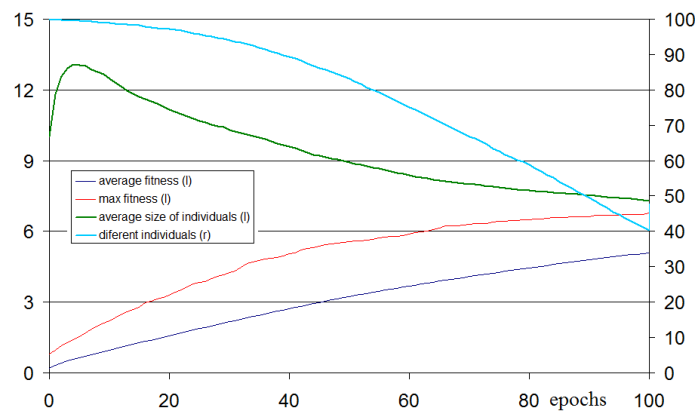
Although the method is promising, the results from this algorithm are comparable to those that used a single objective, but it requires more computational power. Figure 4.6 shows the walk of a population towards the Pareto-front, i.e., the situation where more individuals represent potentially optimal combination of qualities used as objectives for this particular approach. Presumably, assuming better objectives may lead to better results because Pareto-optimization methods are generally known for their efficacy.



(a) 0.01



(b) 0.1



(c) 0.2

Figure 4.5: Steady-state GA evolving run over 100 epochs averaged over 100 runs with different mutation probabilities: 0.01 (a), 0.1 (b), 0.2 (c).

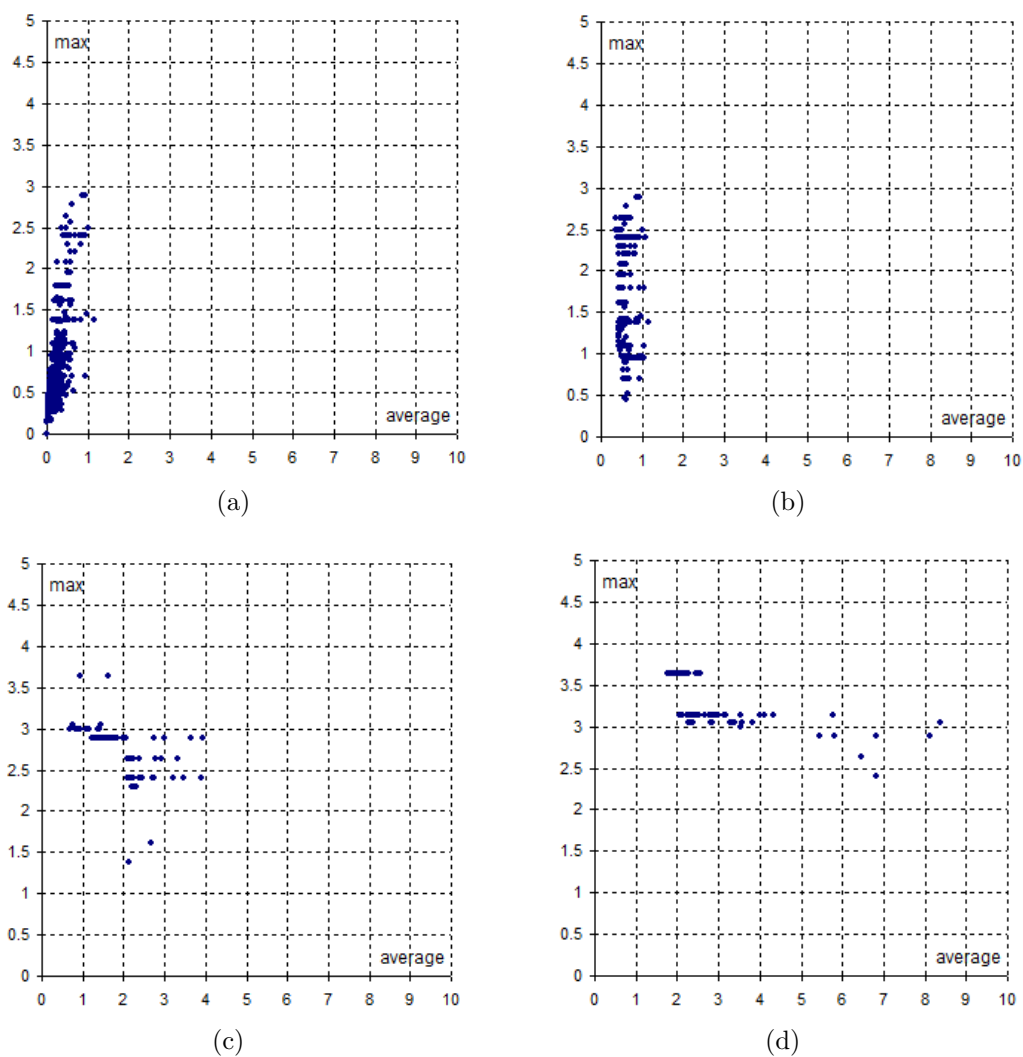


Figure 4.6: Converging towards Pareto-front for music melodies generation at various stages: initial population (a), population after 10 epochs (b), population after 100 epochs (c), population after 1000 epochs(d).








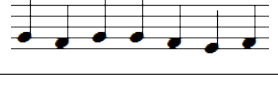

4.3.5 Results

Some melodies generated by the system are shown in Table 4.1. Each melody is preceded by its fitness and the number of runs in which it occurred as a result. They were taken as final results from 100 runs of the steady-state algorithm with mutation since this system set-up produced the best offspring in the most reasonable time. Although this is only a sample of the result set, one can notice that the fitness landscape has some unique properties. There are narrow spikes of fitness surrounded by patterns with a count of 1; and a series of additional ‘hills’ that have a wider base (e.g., the melody with count 15). It is usually easy to put those melodies into a harmonic context and most of them preserve the scale or the key in which they are being played, which is rather surprising, because harmonic context is not directly imposed by our n-gram-based fitness evaluation method. Most of them are in major scale, but some of them are complementary: the melody with the fitness of 8.21 is the major version of the minor melody with fitness 7.98. The important thing to notice is that most of those melodies, even if they have interesting melodic structure, have poor rhythmic pattern separating rhythm from melody. An additional objective will help in solving this issue.

4.4 Issues and Possible Extensions

The method for generating music melodies using a corpus of musical pieces as a guide for evaluating individuals gives encouraging results without resorting to sophisticated evolutionary frameworks. Applying some knowledge of music while formulating the representation has provided the basis for an indirect encoding, thus avoiding the need for specialist search operators. The problem of evolving melodies also demonstrates some new artifacts such as ‘note bloat’ that may be very interesting to investigate. The method proposed in this chapter revealed an interesting area of research in evolutionary computation and may be helpful for the people of music to create new ideas for composition.

Table 4.1: The samples of resulting melodies.

Fitness	Count	Pattern
9.55	1	
8.83	1	
8.46	1	
8.37	1	
8.21	3	
8.05	2	
7.98	1	
7.48	15	
7.17	1	

The results of the approach to date are characterized by quite a simple rhythmic scheme. Separating rhythm from melody may help in obtaining more diversified results, but one cannot make rhythm and melody totally independent from each other. Other music features like establishing a proper beginning and ending and maintaining a common harmonic logic and structure of an excerpt also need to be investigated.

The evaluation of results is still an open-ended issue. Unlike some other previous systems that take feedback from humans for evaluating fitness of individuals, the results from the proposed system do not have any contact with humans until they show up in the final population.

The fitness landscape of the problem for the representation of music and evaluation of individuals presented in this chapter was shown to be multi-modal. There are many local minima, and no information about maximum fitness is obtainable by the system. Moreover, it is difficult to identify useful individuals for latter use. This results in the general contradictory requirement for preserving diversity while maintaining progress towards specific solutions. Allowing melodies to have variable length may lead to the problem dubbed the ‘note bloat’. Can this problem be solved by using some simple form of fitness evaluation that does not force a certain individual length? The Steady-state approach looks like it provides an appropriate mechanism, but on the other hand, most of the final results have a length that correlated with the definition of weights in the fitness function. Addressing all these issues may help in solving similar, real world problems where individuals are organized in a similar way.

There are many other problems such as additional representation issues or application of context-aware search operators. Assuming some basic music theory, we maintain that such problems can be addressed using music-designated solutions. This determines the uniqueness of music-driven tasks. And it means no “free lunch” for those who try to solve music related problems using standard methods and approaches.

Chapter 5

Symbolic Music Visualization

In the following chapter we present a technique for visualizing symbolically encoded music stored in MIDI files. The method is automatic and enables visualizing an entire opus on a single image. The resulting images unveil the structure of a piece as well as detailed themes' leading within it. The technique is suitable for many types of music (both classical and popular) and the quality of the visualization highly depends on the quality of input MIDI file. The program for creating visualizations using this technique and previewing them with MIDI playback is made available for use within the community.

Music visualization systems work with two types of data — raw recordings and various forms of symbolic representations. There are also two different target groups of such visualizations — untutored audiences and musical experts. The former's needs are quite simple — it is sufficient to provide them with a solution that follows the music in some way, enhancing their listening experience. Such visualization systems are present in multimedia players. On the other hand, professional visualizations are designed to convey specific information to users with a proficiency in the music domain, which includes various ways of presenting audio signal for sound engineers. The other approaches incorporate symbolic music representations such as sheet music for music performers to better understand a given opus. The solution proposed here is designed for both music performers to help them understand the structure of a piece, and laymen to track the flow of music. It will also serve as a basis for automatic music structure analysis introduced in the next chapter.

There are many benefits of using symbolic music data for music visualization. Primarily, the data is much cleaner and the actual, high-level music events are clearly distinguished and annotated. As a result, the potential algorithm does not need to

assume that it deals with inaccuracies in the data. Moreover, all formats for storing symbolically represented music allow for separation of logically different voices. This makes it possible to investigate the inter-voice dependencies, which would result in more adequate structure-revealing analyses. This is particularly true for works for music ensembles, where each instrument has to have its own track, according to MIDI specification. However, the most important benefit of using symbolic representation, is a clearly defined local context for each of the music events within that piece. This context is derived from the relationships between these events and other notes within their neighbourhood. Symbolic representation allows for the use of linguistics-like methods of analyzing streams of notes, for example by using text-based methods of determining similarity, as we introduced in previous chapters.

However, there is one problem with these kinds of symbolic-based analyses. Most of the music available and accessible to the general public is in audio form. Proposed methods would work for those cases only if the audio is supplied with annotations describing the low-level music features that are present in MIDI or score files. Current automated transcription systems do not usually work for complex pieces and they introduce too many errors (accuracy reaches from 0.6 to 0.8 depending on the dataset [43]). There are two techniques that could help to overcome this problem. One of them is to supply the audio channel with the actual music event information. Such solution has been proposed [8], but they have not yet become popular. The alternative is to use automated score-audio alignment systems that, unlike automatic transcription systems, are quite accurate. Our solution focuses only on working with symbolic representations, assuming that the alignment or annotation step has been completed.

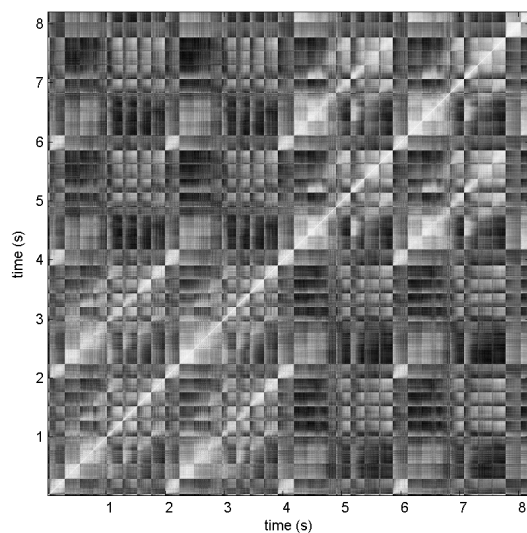
5.1 Existing Approaches to Music Visualization

The concept of similarity matrices was primarily introduced as a mathematical concept of recurrence plots for time series by Eckmann et al. [45]. It applies to any time series data and a result is a two-dimensional image that visualizes internal dependencies within the given time series. This concept, upon which our work is built, has

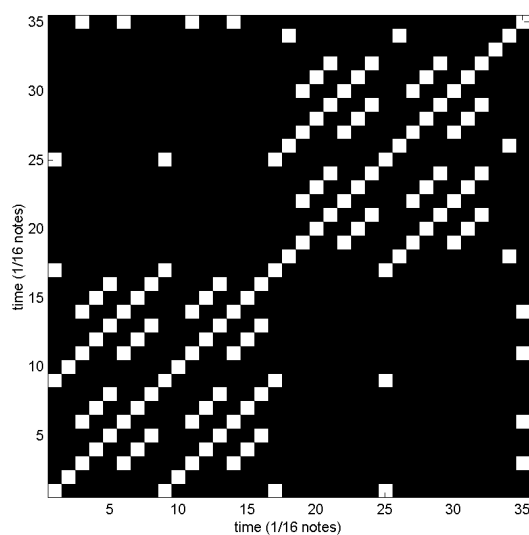
been applied to audio data by Foote [51]. In his approach, a raw music recording is taken as input data. The output is a visualization, that is organized as a rectangular image, where each pixel (at position (i, j)) expresses the audio similarity that results from cepstral analysis of two corresponding excerpts (frames) of the piece (Figure 5.1) at time i and j . Cepstral analysis simulates human perception of audio signals, thus fragments that sound similar for humans tend to have similar cepstral coefficients. Organizing them in a rectangular image allows for tracking of the dependencies in a music piece. Although the techniques summarized in Foote's work were applied to audio-based systems, it is possible to adapt them to operate on symbolic data in a similar fashion.

Using a single channel recording (mono) as input data decreases the complexity of the problem, so there is just one concurrent object to compare in each time frame. However, if we consider music in general, there are usually multiple separated logical channels of music information, at least one for each instrument, voice or stave, which cannot be accurately separated, after these have been blended together within a audio recording. Because of this, many important details of voice dependencies remain hidden. J. Foote presented a sample similarity matrix that results from analyzing a MIDI file (Figure 5.1b), but his simplifications in this area remain significant: one channel with one note compared at a time. Symbolic representations hide all the performance-dependent features but carry the entire structural information. Incorporating this information is one of key benefits of this kind of data. This can result in more complex and sophisticated analysis [9, 121, 172].

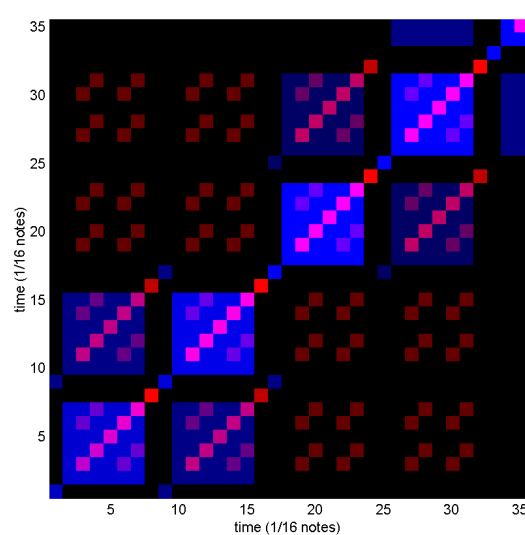
Isaacson [85] in his introduction to various music visualization systems, points out, that audio “has many limitations as a representation for analysis. Humans have the (remarkable) ability to recognize individual components in a sound source, including identifying specific instruments ..., as well as melodic lines and rhythmic patterns within each, and to translate that information into a mental symbolic form that is more reminiscent of the musical score than of a spectrogram. ... It is exceedingly difficult to extract this information from an audio signal, and hardly more visible in



(a) audio MFCC



(b) symbolic binary



(c) proposed scheme

Figure 5.1: Bach prelude C major — an excerpt visualized using three methods: a) Foote [51], b) symbolic based on exact notes comparison, c) with the proposed context-aware approach

a picture of that signal. In fact, except when spectral (i.e., timbral) information is specifically the focus, the visual ‘noise’ that the overtone structures add to the image masks much of the information that is traditionally of interest in music analysis” [85]. Symbolic representation is a more suitable source material for high-level analysis even though incorporating it leads to the problem of conversion from audio to symbolic music. This was shown in the paper describing the ImproVis system [172] where the manual transcription of recorded performances practically prohibited a wide application of the presented method.

The most practical approach is therefore the visualization of existing, symbolically coded music, such as MIDI files, that incorporate MIDI protocol to encode musical events such as notes, rests, etc., and require much less disk space than audio files. However, most of the existing MIDI visualization systems either simplify the visualization problem by just modifying western music notation in order to add some other visual features (colour, line thickness) and displaying them on the score staves [121], or extract from the music some very sophisticated (e.g., harmonic) features, visualizing them in a very specialized way for a very specific purpose, so not practical in general.

One of more sophisticated approaches was presented by Bergstrom et al. [9]. The approach presented in that paper visualizes tone classes instead of notes, which reveals the harmony structure of a given piece of music. Tones are placed on a grid, based on Tonnetz template, such that notes belonging to a chord form a cluster. Similar chords lie close to each other on this grid, making chord progressions easy to observe. The method can be used in music education for teaching harmony rules or may aid professionals in their work with music. On the other hand, using this kind of tools is strictly limited to people with a certain proficiency in the domain of music theory so even self-taught musicians without theoretical background may not find this visualization technique helpful.

A solution utilizing a similar concept of visualizing symbolic music using self-similarity matrices for monophonic symbolic music, developed in parallel to our solution, was developed by Hanna et al. [71]. They also used melodic intervals and IOR features to accommodate for transposition and tempo invariance, as well as local contexts. They used editing algorithm to compare contexts, which takes care of small variations in melodies, while masking fine textural information. Main limitation of the solution was its strict requirement of the monophonic music as the input.

5.2 Methodology for Symbolic Music Visualization

Our solution to visualizing symbolic music tries to address the problems of existing visualizations of symbolic music. This tool does not need any pre-processing steps. It takes MIDI files as input and visualizes the content of those files so the quality of visualization depends on the quality of the MIDI file, where the quality is determined by the approach to sequencing a MIDI file. One can imitate the original score, preserving voicing structure in channel structure, preserving notes durations indicating note lengths, and so on. But MIDI can also be used to encode a performance on an electronic instrument in a very compressed way. In this case there is usually one track with a mass of notes with timing taken from the performance that may include augmentation and diminution of tempo, and where notes do not have to be aligned exactly. The resulting MIDI file sounds more realistic but is not suitable for analysis and, to some extent, for this kind of visualization. Fortunately, there are relatively few instances of such MIDI files, as most available MIDI files that can be found in various repositories, such as *classicalarchives.com*, come from score transcriptions.

The presented approach allows for wide parameter customization so that the user can filter and limit what they think should be visualized. The entire visualization system is available as a program that allows previewing and playing back of the underlying music piece enabling users to track the visualization along with audio.

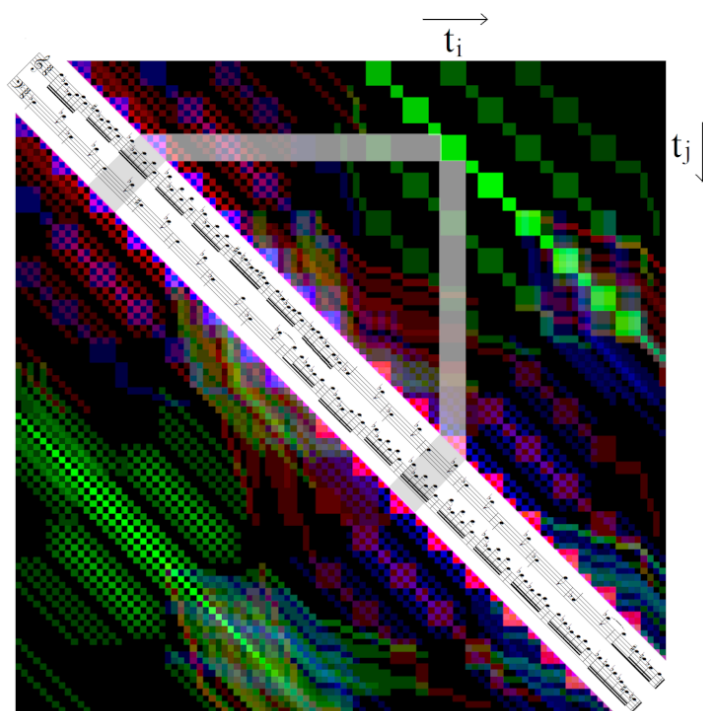


Figure 5.2: Visualized excerpt from Bach prelude C sharp Major

5.2.1 Use of Similarity Matrices as a Layout for Visualization

In order to visualize music stored in MIDI files, we generate similarity matrices that correspond to Foote's system [51]. They incorporate symbolic music representation as input data. Since symbolic music is more complex in its structure compared to the raw recordings, there are many problems that emerge which will be addressed in this approach. Although performances have at least the same level of complexity as a corresponding musical score, everything is merged together into a single waveform (i.e., one dimensional, time dependent function) and it is almost impossible to recover the underlying high- and low-level structure using techniques currently available. In our solution, a visualization is organized in a rectangular colour image, where each square in the image represents the similarity between two corresponding notes that are played at two corresponding times, t_i and t_j (Figure 5.2).

Time proceeds from the upper left corner to the lower right. The diagonal line shows the currently played notes as very bright since self-similarity of any excerpt is always assigned the highest possible value.

5.2.2 Dealing with Multiple Layers and Tracks

MIDI files, as do their underlying musical pieces, consist of many concurrent sources of notes. In the corresponding digital audio signal approach, this problem never arises because there is only one currently played track of music at a time. In real music, one has many logical channels with each representing a different hand, a different instrument or a different voice. The question is, how to incorporate those inter-track dependencies? In Foote's proposal, the resulting images are greyscale since just one feature has been visualized (audio is one-dimensional), and hue remained free for visualizing other features.

In our solution we reserve colour space to show dependencies between different music tracks. For example, in Figure 5.2 blue colour represents self-similarity of the left hand, red colour shows the self-similarity of the right hand and green shows the similarity between left and right hand. Larger blue squares at the first half of the diagonal axis and red ones in the second half represent longer notes played in the corresponding voices. The fine checkerboard pattern of the red small squares at the beginning (the top left corner) and the same pattern of blue colour around the lower part of the axis show the self-repetitive fine structure of the 16ths' groups in both hands. Green patterns indicate inter-voice dependencies. Larger squares in the upper-right corner show the repeating pattern of quarters and eights that are transferred from the left hand to the right hand. The fine checkboard structure of the green cloud in the lower left corner shows the fast pattern of 16ths that moved from the right to the left hand.

The other problem is how to handle chords and concurrencies in a single track. One can not easily determine what is the direction of the melody if one finds a chord. According to previous research [193], it has been shown that one concentrates on the

highest currently played note so the simplest approach for dealing with concurrencies is to retain only the highest note at a time frame. This approach to dealing with concurrencies has been used in our system.

5.2.3 Comparison Function for Music Excerpts

Assuming that we now have a linear note structure in every track and we compare each pair of tracks, one has to define a certain music representation and comparison function. The trivial approach to this problem is to take notes lengths and pitches and map them directly to a comparison function. As it was shown in the previous sections, this approach has two main caveats. It does not preserve melody direction while comparing two similar melodies that lay on different pitches or are played in different scales. It also does not preserve rhythmic similarities that exist while the same melody is played slower or faster. Moreover, the same eighth note with a certain pitch may mean different things in two different excerpts since its role depend also on the neighbouring notes.

These issues may be overcome if one takes relative pitch and relative duration (melodic intervals and rhythmic inter-onset interval ratios (IORs)) to form basic features, unigrams. The same melody played in various tempos and in different pitches gives the same sequence of unigrams. This is especially important in analysing fugue themes, which occur on different heights and may be played with different paces.

The last issue to be addressed in this section is how to determine the similarity of two sequences of unigrams. The solution proposed is that the similarity is the number of the same (overlapping) unigrams in both excerpts within a certain window:

$$sim(\{a_i\}_{i=1}^N, \{b_i\}_{i=1}^N) = |\{i \mid 1 \leq i \leq N, a_i = b_i\}| \quad (5.1)$$

where $\{a_i\}$ and $\{b_i\}$ are two input sequences, and N is the window size. Having windows of size N will result in similarity levels varying from 0 (no similarity) to N (full similarity). It is then encoded by the saturation of a certain hue on the resulting image. The choice of colours does not matter at this point, but primary colours

are preferred due to the mixing problem described later. The values from different layers are then summed up and normalized across the image. Figure 5.1 and Figure 5.2 show in details how the level of brightness is applied. However, both presented excerpts do not have any problems with colours assignment because they both have only two channels which gives three possible colour layers. Keeping in mind that the basic colour table has three dimensions (i.e., red, green and blue) three layers can be displayed without ambiguity.

5.2.4 Dealing with Layer Overload

The number of layers grows quadratically with the increasing number of visualized tracks. Adding one more channel to the simple pair requires six combinations, which with the support of the secondary hues (cyan, magenta and yellow), can also be visualized. However, in this situation some overlapping similarities, e.g., on the red and the blue layer, may be misunderstood as a similarity on the magenta layer. The situation becomes more complicated if one has even more channels. Figure 5.3 shows the excerpt of Bach's Fugue C# minor with 5 voices (15 possible layers), where we had to repeat certain hues or different channel pairs.

Figure 5.4 shows just these layers that compare with the 3rd voice. The amount of information is smaller but one can easily observe the theme (saturated green pattern at the beginning) and fine structure of counterpoint (middle of the image) repeated in remaining voices (high similarity to other voices).

Filtering The other important feature of the visualization is that it enables filtering by a set of unigrams. In this case, the system will increase the similarity of two excerpts only for the unigrams in those melodies from a fixed, predefined set. This allows content based visualization of certain types of melodies. Figure 5.5 shows the same excerpt as in the Figure 5.3 with respect to the unigrams present in the theme of the Fugue. One can observe the yellow theme at the beginning of the excerpt that corresponds to the green one (different colour assignment) in Figure 5.4.

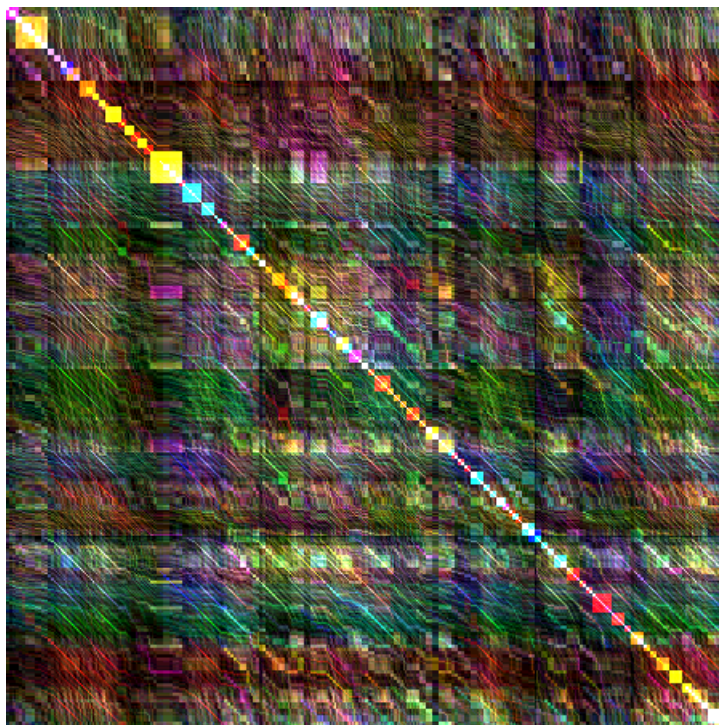


Figure 5.3: Plethora of layers in 5 voci C sharp minor Bach's Fugue. Middle of the piece. 15 layers.

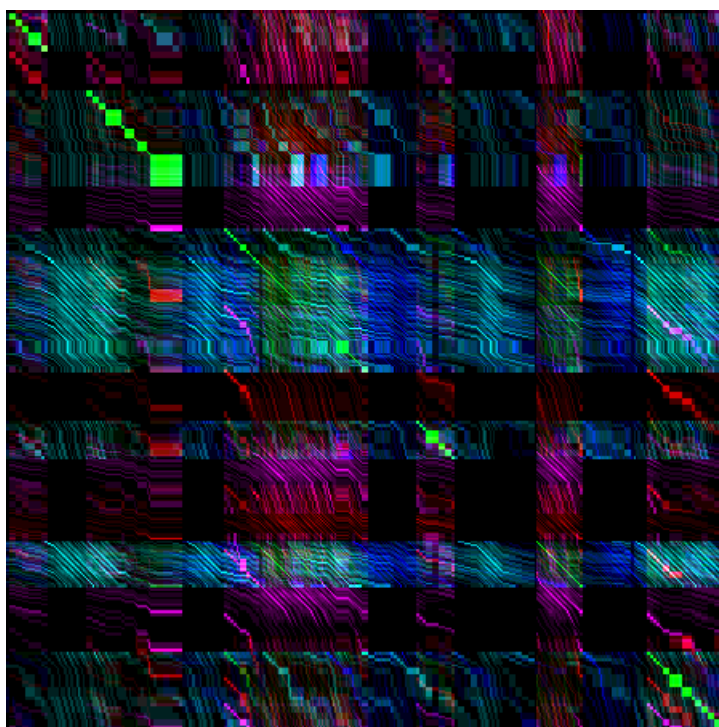


Figure 5.4: Excerpt from Figure 5.3 displayed 3rd voice comparison only (1&3 – cyan, 2&3 – blue, 3&3 – green, 3&4 – red, 3&5 – magenta).

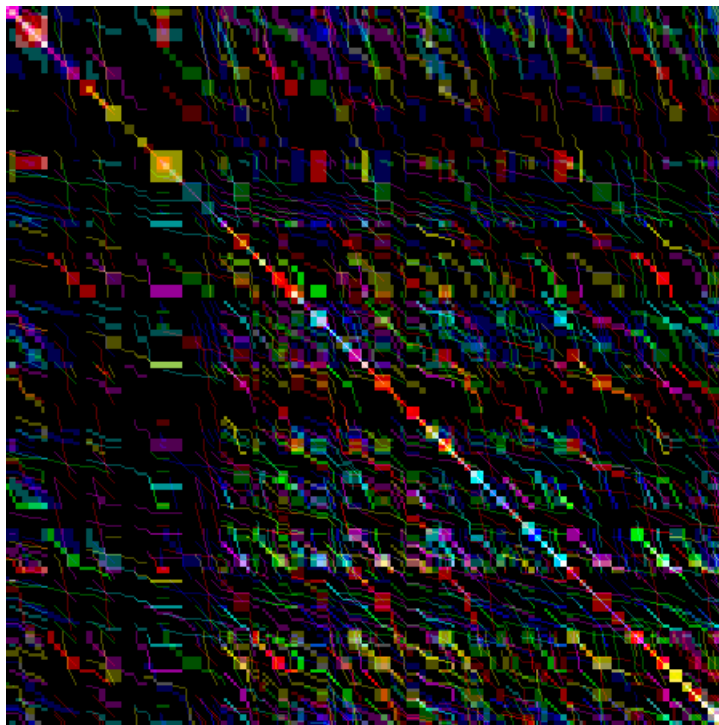


Figure 5.5: Excerpt from Figure 5.3 filtered by the theme.

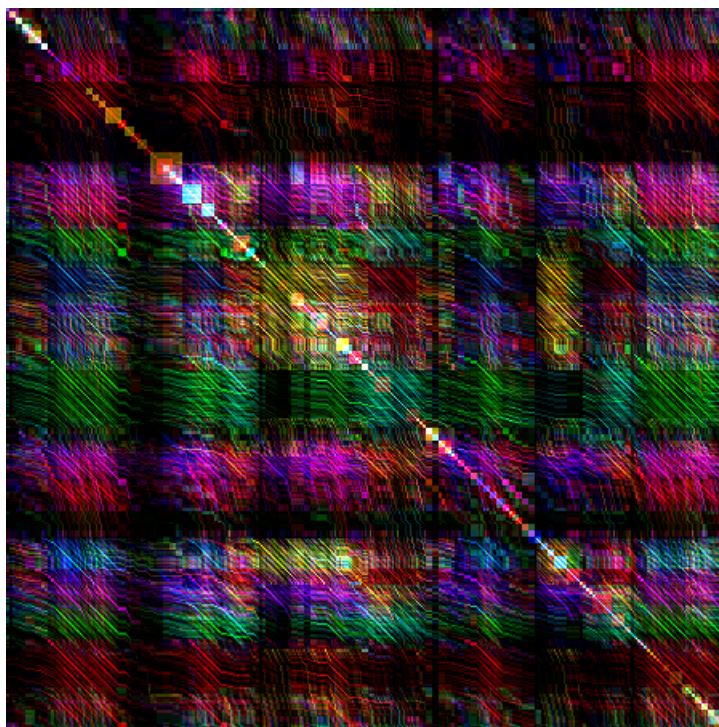


Figure 5.6: Excerpt from Figure 5.3 filtered by the counterpoint.

Figure 5.6 presents the same excerpt filtered with the unigrams characteristic for the counterpoint of the theme. The theme is easily distinguishable from the counterpoint because they demonstrate quite different rhythmic and melodic styles.

5.2.5 Algorithm for Matrix Generation

Algorithm 1 presents the general procedure to obtain self-similarity images. A variation to it, presented in Algorithm 2, is more suitable for generation of small, not full-size images. These two approaches feature the same computational complexity, and the only difference in their performances lies in sizes of generated images. The first one iterates over each pair of notes and sets many pixels at once, corresponding to the respective area on the image. The second version iterates over each pixel in the image, assuming it would require less similarity computations, as not all notes will be involved in generation of a small thumbnail.

The most precise result occurs when even the shortest note appears in the final visualization. This makes images very large, and longer notes contribute to many pixels simultaneously, which makes Algorithm 1 a default choice.

5.3 Visualization Package

The visualization program was written in Perl with a graphical user interface. The application was then compiled to an executable file using Perl::Packer module and the NSIS installer was created for the application.

The application opens with an empty window waiting for a MIDI file to be chosen. After the MIDI file is loaded the user sees a window with the following controls:

- A. A visualization panel with the visualized piece in the background and two sliders for playback control.
- B. The status bar informing the user about system state and showing the progress of operations.

Algorithm 1 Symbolic music visualization I

```
1: determine the size of the visualization based on the shortest note from the MIDI
   piece
2: for every track1 in MIDI do
3:   for every track2 in MIDI do
4:     colour = DetermineLayerColour(track1, track2)
5:     for every note1 in track1 do
6:       for every note2 in track2 do
7:         area = IdentifyArea(note1@track1,note2@track2)
8:         similarity = CalculateSimilarity(note1@track1,note2@track2)
9:         AdjustLayer(area,similarity,colour)
10:      end for
11:    end for
12:  end for
13: end for
14: SaveImage()
```

Algorithm 2 Symbolic music visualization for thumbnails

```
1: set a priori the size of the visualization
2: for every  $y$  in  $Image.Rows$  do
3:   for every  $x$  in  $Image.Columns$  do
4:     for every  $track1$  in  $MIDI$  do
5:        $note1 = IdentifyCurrentNote(y, track1)$ 
6:       for every  $track2$  in  $MIDI$  do
7:          $note2 = IdentifyCurrentNote(x, track2)$ 
8:          $colour = DetermineLayerColour(track1, track2)$ 
9:          $similarity = CalculateSimilarity(note1@track1, note2@track2)$ 
10:         $AdjustLayer(x, y, similarity, colour)$ 
11:      end for
12:    end for
13:  end for
14: end for
15:  $SaveImage()$ 
```

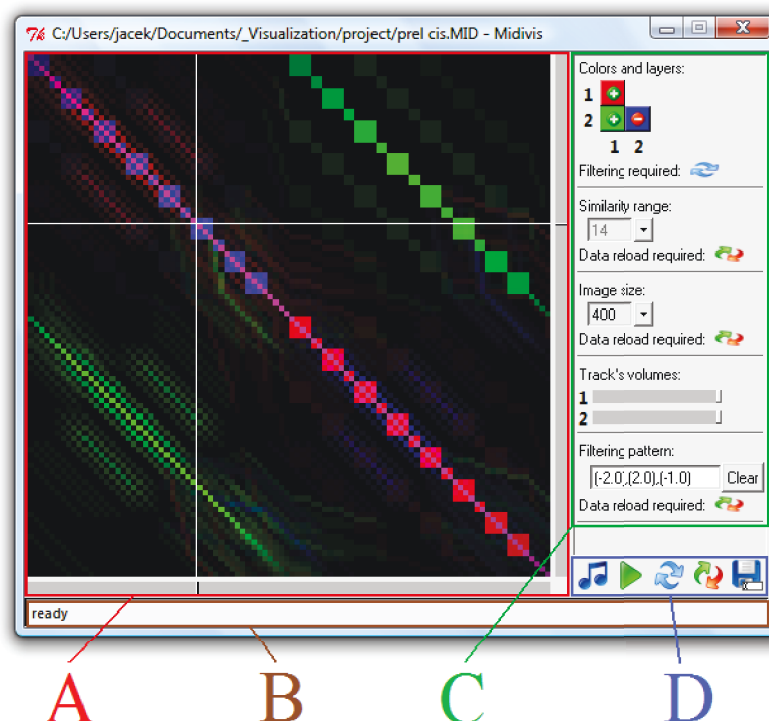








Figure 5.7: Program main window with the following components: A) visualization pane B) status bar C) visualization controls D) toolbar

C. Visualization properties and visualization dependent controls. Each field contains an indication if changes in a following section require refreshing an image by simple reviewing image data (🔄) or require reload of the whole file, which usually takes more time (🔄). Controls include:

- (a) Colours and layers controls. Colour of a box describes the hue of points representing similarity on a given layer. Symbols on the buttons represent if a corresponding layer is to be included in the final visualization (⊕) or not (⊖). Right-clicking on a corresponding layer button allows changing the hue of the layer, while left-clicking controls the layer toggle.
- (b) Similarity range — the range (a window) of unigrams within the similarity is measured. The possible values vary from 2 to 14 with a step of 2. Larger values create more ‘smooth’ visualizations. Smaller values produce ‘angular’ visualizations faster.

- (c) Image size — the size of the image presented in the visualization pane. Since source MIDI files produce usually much larger and thus time-consuming images, the adjustment of size can be done efficiently once, at visualization creation time.
- (d) Filtering pattern — unigrams that are chosen as a filter of the image. User can point within the visualization image, and with a right click, get those unigrams that give the highest similarity among all active layers in this point. User can edit those unigrams manually in a textbox at any time.
- (e) Track's volumes — volume controls for playback, separate for each track so that the user can listen the piece focusing on certain tracks. The volume can be changed any time, even during playback.
- (f) Playing speed — the speed of playback. The change during playback does not affect current speed, until it is paused and un-paused.

D. Toolbar containing the following buttons:

- (a) Load new file () — to load new file and open it with default visualization properties
- (b) Play/Pause ( / ) — to start/play playback. It will start in the moment pointed by white indication lines in the visualization pane, which can be controlled with a mouse.
- (c) Refresh () — to refresh current view according to changed settings that do not require reload of data.
- (d) Reload () — to reopen the file again keeping all the settings already set. Required for most parameter changes to update the visualization.
- (e) Save image () — save the current view with optimal image resolution based in the analysis of the content. The images are stored in a lossless PNG format and usually do not exceed 1MB.

5.4 Examples and Features of Proposed Visualization Scheme

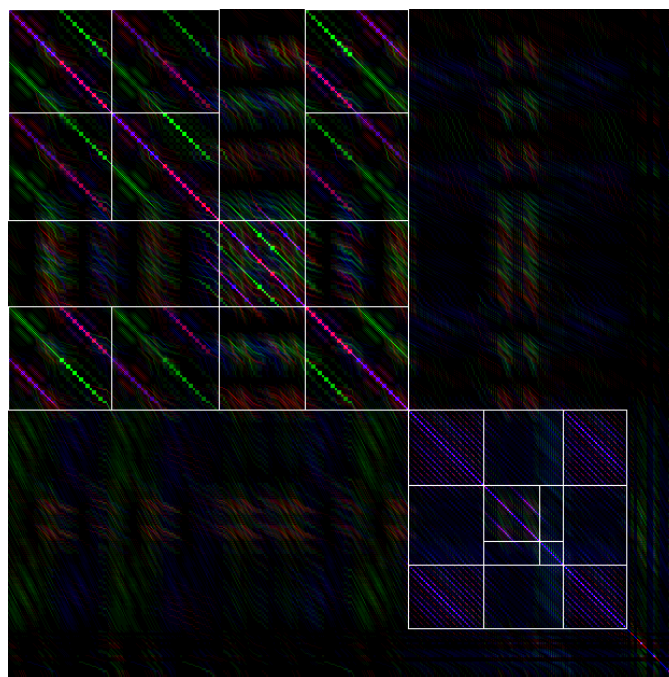
The visualization technique proposed in this chapter unveils the structure of a piece. Similar sections containing similar melodies will occur as repeating squared blocks on the diagonal for every such section and on the side — which indicates the repetition of a section. Moreover, repeating themes occur in the visualization as graphical patterns drawn parallel to the diagonal axis if a melody occurs in two different places.

This technique seems to be especially suited for polyphonic music. Polyphony is a kind of music with two or more independent voices (voci), all leading their own, equally important melody lines. They usually contain similar themes that are moving between voices throughout the piece. The next interesting aspect of polyphonic music for this visualization technique is that it does not contain many chords, which have to be simplified to a single note in our approach, so less information is lost.

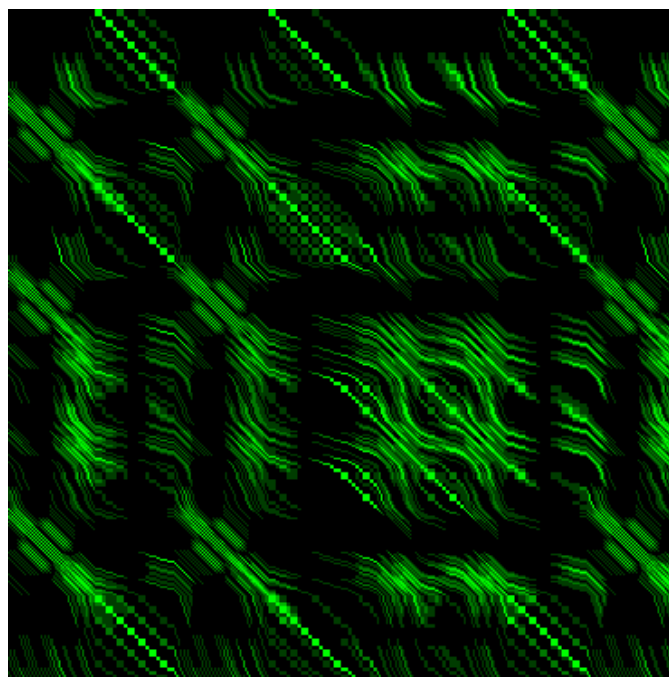
Figure 5.8 shows Bach's Prelude C sharp major with indicated structure of a piece. One can observe two passages of music phrase followed by short intermezzo and then followed by the repetition of main phrase but with inverted roles of right and left hand. After them, there is another, very different section with high frequency patterns, which appear as a thin long lines close and parallel to the main diagonal (to the melody). If those lines are close to the diagonal it means that the melody represented by this line repeats with a very short period.

One can also see the magnified part of this prelude with repeating sections shown in Figure 5.8. The inter-hands layer was the only layer visualized. One can observe how the coarse theme that primarily occurred in the left hand moves to the right, while the fine structured theme (checkboard pattern) moves from right to left hand. This pattern is repeated many times. In the intermezzo, one can observe how the melody moves from one hand to another by the waving patterns followed by repetition of the main theme.

These repeating themes are especially important in fugues. Fugues are a polyphonic genre with a specific, hierarchical structure (parts and subparts) and its main goal is to convey the main theme through voices in the whole piece. The main game



(a) piece structure



(b) voice dependencies

Figure 5.8: J.S. Bach prelude in C sharp major. **Upper picture:** highlighted the visible hierarchy of the piece (manual marking), left hand has blue colour, right hand has red colour. The inter-hands relations are encoded in green. **Lower picture:** The magnified first part of the piece with only green layer displayed. Transitions of themes between voices are clearly visible.

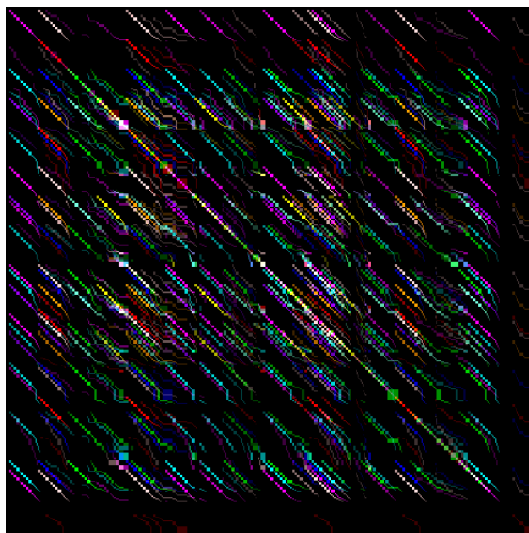


Figure 5.9: J.S. Bach — Fugue in C major. Filtered by the main theme pattern. Unveils the number of themes in the piece.

between the composer and the performer is that the composer tries to hide as many themes as possible and the performer has to find and emphasize all the theme melodies so that the listener, the third participant of the show, can clearly recognize them. The tool described in this chapter should help players and listeners to better understand composer's intentions.

Figure 5.9 shows the Fugue in C major filtered by the pattern collected from the first theme occurrence. One can see how many theme instances were packed in this fugue by Bach.

This visualization technique also shows modified themes. The composer may augment or diminish a theme. However, the unigram structure of this theme differs only by the first unigram, so it still remains similar to the original theme. The only difference will be that it will not be parallel to the axis but rotated. Figure 5.10 shows one theme repeated three times with different speeds. The augmentation and diminution is clearly visible.

Figure 5.11 shows the structure of fugue in E flat minor with everything except the main theme sieved out. One can observe that some themes in the last part of the fugue were augmented.

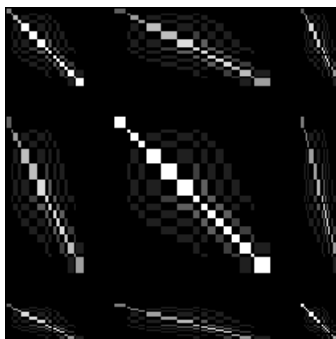


Figure 5.10: Three occurrences of the same theme, played with three different tempi.

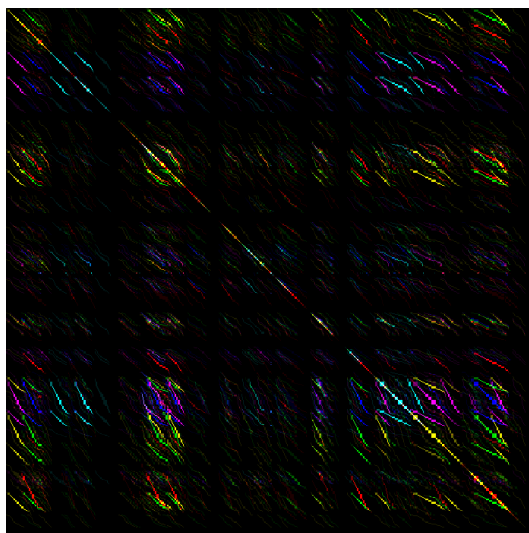


Figure 5.11: J.S. Bach — Fugue in E flat major. Filtered by the main theme pattern. Augmented themes are visible in the second part of the piece.

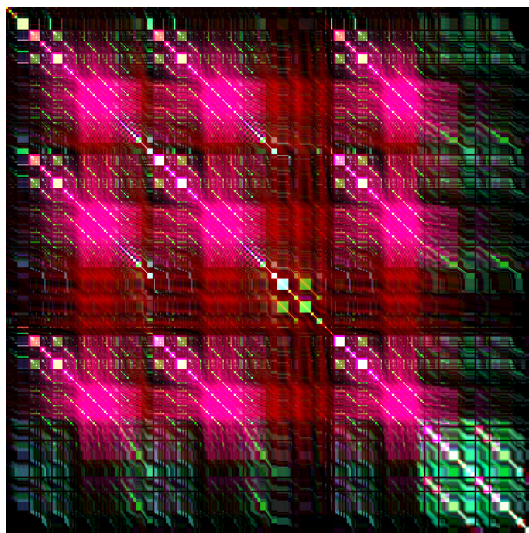


Figure 5.12: The Beatles — The Magical Mystery Tour. All melodic layers visualized. The structure of the piece is easy to perceive

This visualization technique is suitable not only for polyphonic (style) and classical music. Figure 5.12 shows the visualization for the Beatles' song "The Mystery Tour". The structure of three verses with choruses, an intermezzo between second, and third verse and the ending coda are clearly visible. The thin light lines in each verse unveil repeating melodies within the verse. The high brightness of the image shows significant similarity between sections, which indicates the simple structure of a piece (the repeated themes are just copied, so the similarity reaches highest levels). Regarding popular music, one has to keep in mind that they typically contain one or more percussion tracks — they are encoded using the same MIDI protocol, but the pitch value in those tracks does not indicate pitches of notes played, but the instrument used to play the note. Therefore those tracks are not meaningful and are removed from the visualization for its clarity. In this case, the last track of the piece was the percussion track and it has been removed from the final visualization.

5.5 Future Work

The technique of visualizing music represented symbolically in MIDI files presented in this section seems to be especially suited for music performers to aid them in better understanding the piece.

The algorithm was analyzed on polyphonic baroque music, but it should also work with other kinds of music, even modern, and in many cases it will reveal their simplicity and highly repetitive nature. Music enthusiasts may find it appealing, that their music player reveals the structure and theme leading, while playing their favourite songs. Keeping in mind that automatic audio-to-notes transcription seems to be unreachable at this stage, if the audio files are enriched with the symbolic content before the distribution of an album, one can use this rich information for more advanced visualizations of audio tracks.

The technique enables the production of the images of any size, and the smaller images require much less computational time and resources, the generation of MIDI thumbnails is then possible. Since the power of personal computers increases in a rapid pace, thumbnailing, that is, previewing the content of a file in the file icon, has become more and more popular. Using the technique presented in this chapter, one can produce thumbnails of MIDI files and other kinds of music files that contain symbolic information (such as suggested enriched audio files, or score files). The efficient implementation of the program for generating these thumbnails is left for future plans.

The system was briefly reviewed by music experts and they agreed on the capabilities of the system, but a thorough evaluation through a user study should be the part of the future work. It is possible, however, to use this visualization approach as a foundation for other tasks. Then, next chapter shows an extension of this technique which automatically analyzes structure of symbolic music.

Chapter 6

Structural Analysis of Symbolic Music

Automatic discovery of patterns and structure in music is a very active area in today's computational musicology. Most of the approaches to this task focus on analysis of audio data, while some were indicating that it could be beneficial to apply those methods to music represented symbolically. The proposed automatic structure analysis system is based on creating self-similarity matrices and implements a text-like n-gram representation of symbolic music data. The aim of this work is to explore how various similarity measuring techniques adopted from Information Retrieval and Natural Language Processing help in exploring structure of symbolic music. We have found that the context of each music event within a piece plays a very important role. The modified Dijkstra algorithm is then applied to the matrix, revealing its structure and patterns derived from the obtained shortest path. The proposed solution has been evaluated on both classical and popular music, showing significant improvement in both precision and recall.

6.1 Motivation

Structure is a very important aspect of music. Studying it is a vital part of musical analysis which unveils human cognitive processes upon creation of a given music piece. Being able to automatically unveil this structure through a computer program would mean that one can analyze this aspect of art in an algorithmic way.

One can point out a number of applications of music structural analysis, including active music listening and navigation [16, 191], music summarization [25], automatically locating certain sections [63], a framework for semantic music analysis [117],

music creation [134], playlist generation [2], improving audio based music information retrieval [88] or aiding research in ethnomusicology [91].

Structure in music is driven by repetition: repetition of certain parts and themes, but also repetition of music textures, or rhythmic and harmonic patterns. According to Schenker, “repetition is the basis of music as an art” and repetition leads to structure of a piece. Thus, in order to determine structure of a piece of music one has to mine the internal dependencies and repetitions within that piece.

Mining of self-similarity matrices has become recently a very popular strategy for structural analysis [28,147]. It is a very convenient way of visualizing music structure because, if the feature extraction and comparison methods pull relevant information, the structure is usually clearly visible on a resulting 2-D image. However, it has been used only in audio-based systems, despite the concerns that symbolic representations carry the actual information that similarity based audio comparison functions actually try to identify [28,125]. Indeed, the simple, same-or-different binary comparison approach applied to note data does not typically result in a sophisticated structure visualization (see Figure 5.1(b)), but for more advanced approaches, presented in this chapter, the structuring becomes quite clearly visible.

In this chapter, we propose our solution to the problem of inferring structure from music pieces represented symbolically. Since most applications in this area operate on audio data, we assumed that either symbolic annotations are available or one obtains them through some other preprocessing steps, like automatic transcription or score to audio alignment. To fully describe the structure of the piece, the solution should find the number of sections the piece divides in, place them precisely within the piece, and identify and name them, e.g., using symbols like A , B , C indicating similar and different sections. Since there are currently no solutions that operate on symbolic representation, the proposed algorithm will be tested against the existing approaches operating on audio data. The evaluation will be based on precision and recall of correctly assigned and placed section borders. Most of the existing test beds

do not evaluate the quality of determining relationship between sections (i.e., naming the section with symbols), so we will omit this aspect in our quantitative evaluation.

6.2 Related Work

Most of the existing literature, that deals with music structure analysis, focuses on audio data. Works dealing with symbolic representations are rare, but they exist. Murphy [134] proposed an algorithm to mine for low-level patterns on a phrasing level in MIDI files. Cambouropoulos [19] analyzed approaches of identifying repetitive segments in symbolically represented pieces. Hanna et al. developed a method of obtaining self-similarity matrices for monophonic symbolic music [71]. Dannenberg and Hu [29], and Marolt [125] tried to infer higher-level, symbolic features from low-level audio and then infer the structure based on this mid-level representation. However, most of the existing work in music structure analysis is focused on audio data and similarity matrices.

The use of similarity matrices, introduced in the previous chapter, has been widely used for audio-based music structure analysis since Foote's publication [51,52], which was summarized by Dannenberg and Goto [28], and Paulus et al. [147]. Numerous systems were proposed using different ways of measuring similarity between two excerpts, mainly using MFCC (mel-frequency cepstral coefficients), chromatic (i.e., spectral) or rhythmic (i.e., temporal) features. Paulus describes 28 such audio-based structure analysis solutions, with a number of them focusing on the analysis of similarity matrices. Among them one can distinguish three main approaches to the problem:

1. Novelty-based, whose aim is to automatically identify points of change in the audio, in order to detect borders between sections.
2. Homogeneity-based, whose aim to identify homogeneous sections of a piece. This is an opposite approach to the one used in novelty-based methods.

3. Repetition-based, whose focus on detections of repetitive sections within a piece. This allows for structural grouping of similar sections that may repeat in different places within a piece.

The proposed method for symbolic music structure analysis uses both novelty and repetition paradigm. However, the end result depends only on repetition of certain fragments (novelty is used only to precisely place section borders only, when a repetitive fragment is detected). A drawback of this kind of approach is that mining solely for similarities may lead to the situation where a part that does not reoccur, may not be revealed [146]. Birmingham et al. [13], identify structural analysis with detection of repeated patterns. In their proposal of new techniques for music information retrieval, they suggest using notion of recurrent state as it “abstracts the concept of structural organization in a piece of music” [13].

Dannenberg and Hu [29] presented three algorithms, operating on transcribed monophonic audio, transcribed harmonic progression and spectrum-based analysis. For symbolic matching they used direct note equivalency (same pitch, same duration). They also recognized that simple application of dynamic programming would lead to trivial solution of one part spanning the entire matrix across its diagonal and they proposed heuristic techniques to identify stripes as section repetitions. Goto [63] suggested a different intermediate representation — a time lag plot, where repeated segments appear as horizontal lines. He used it then to extract the part that occurs the most to mark it as a chorus. Similar technique of extracting patterns operating on similarity matrices was proposed by Peeters [149].

Marolt [125] also pointed out that incorporating higher-level symbolic information would be beneficial in structural analysis, and like Dannenberg and Hu [29], employed melody and rhythm extraction from audio. Similarly to our previous approach to visualization, they used sliding window to incorporate context in similarity calculations and they also used cosine similarity measure. However since they used absolute values for pitch and duration, their approach would not be tempo, transposition and

progression invariant, which may not be that important for pop music but play important roles in classical music analysis. Their context size is quite large (above 10 notes), which indicates that the resulting analysis will focus on melodic phrasing, rather than melodic texture. They suggested that using this higher-level information would outperform typically used audio-based features, like chroma or MFCC. Although they did not advance their analysis beyond similarity matrices, they used the results of theme extraction for information retrieval purposes.

Goodwin and Laroche [62] incorporated dynamic programming to determine partitioning of a piece where we stay in a state if the distance to the centroid is kept low. The algorithm uses dynamic programming to determine the best segmentation, as they included two kinds of costs — local, as an indication of staying in the same state, and transitional — for jumping to the new section. Their focus was to group similar frames together, so if a part consist of radically different areas, it will be separated into many parts. Jensen [90,91] used a similar, dynamic programming approach to find the best partitioning among all possible partitionings of the piece. Although both solutions use dynamic programming, their approach is very different to the one presented below, as they traverse through the graph of partitions as the way to find the best partitioning, while we are traversing through a self-similarity matrix and the structure analysis is inferred from this path. Similar approach could also be seen in Peeters [148].

Muller and Kurth [133] suggested a two-step structure analysis process, where one finds and isolates off-diagonal stripes on the similarity matrix (stripes indicate section repeats) and then apply structure analysis process to find the description. What is worth noticing is that they introduced stripe pruning mechanism to remove unwanted patterns based on graphical features of the obtained similarity matrix. We suggest how one can do it in the preceding step, upon similarity matrix calculation, based on original vector features.

Paulus and Klapuri [146] described the system that creates a full description from an audio stream. The entire piece is initially segmented with border candidates (30

of them, arbitrarily for each piece) using Foote’s boundary detection kernel [51] and then partitioned to form descriptions. The objective function of this greedy approach is to maximize the fitness of the description, based on the similarity of segments. A similarity of two segments depends on the shortest path through the subblock of the similarity matrix bounded by those two segments. The best partitioning is found by a version of a token-passing algorithm that browses the space of possible partitions.

The closest approach to the one presented here can be found in Shiu et al. [169], where a similar matrix traversing approach was introduced. However, they forced the path to stay off the diagonal so the resulting path would not indicate all the section borders, leading to an incorrect description if our approach was taken. They used standard novelty detection approach to find section borders and the shortest path through the similarity matrix serves only as indication of relation between sections.

In general, the existing solutions either focus on just one part of the process or create the entire structural description in many disjoint steps, using different methods in each step (e.g., calculating self-similarity matrix, finding borders, defining similar sections, grouping/naming sections). We propose a two-step model: after finding the similarity matrix, the pathfinding algorithms finds all the boundaries, defines similar sections and describes the piece at once. This makes the solution less expensive than other solutions that incorporate dynamic programming, typically with $O(n^4)$ (e.g., Jensen [90] or Peeters [148]), while our approach requires $O(n^3)$.

6.3 Methodology

One of the most important benefits of using symbolic information is the ability to link musical events together to utilize dependencies between them. This differs from mining in audio-based similarity matrices as proposed by Foote [51] where each point on the similarity matrix depended only on the corresponding fragments used for calculating similarity. As was pointed out by Paulus et al. [147], although using small windows leads to finer and more accurate similarity matrices, longer window sizes and coarser images are better for determining the structure of a piece.

The general idea behind creating similarity matrices based on symbolic music consists in determining similarity between each pair of notes in the piece and organizing them in a rectangular matrix (Figure 6.1), which formally could be denoted as:

$$\mathbf{S} = [a_{i,j}]_{M \times M} \quad \text{and} \quad a_{i,j} = \text{sim}(x_i, x_j | \mathbf{c}_i, \mathbf{c}_j) \quad (6.1)$$

where M is the number of notes in the piece, the similarity function sim is any function of four arguments, where x_i and x_j represent notes at positions i and j respectively, and \mathbf{c}_i and \mathbf{c}_j are contexts, which are vectors of dimension N . They could be more precisely expressed as: $\mathbf{c}_k = (x_{k-\lfloor N/2 \rfloor}, \dots, x_{k-\lfloor N/2 \rfloor + N - 1})$.

Since there are usually multiple channels of notes that are played at the same time, in order to find dependencies between all voices, such a similarity matrix can be calculated for every pair of voices. We called them layers and we have used them previously in music visualization [209]. With n voices it gives $\binom{n}{2} = n^2/2 - n/2$ possible layers. At this point, there is no reason to merge them for analysis purposes. It is fine to keep them as a multi-dimensional feature vector for further analysis, as it was used in the previous chapter for visualization purposes, where the number of dimensions of the feature vector was reduced from $\binom{n}{2}$ to 3 for red, green and blue channels. For the analysis of the influence of various similarity measures we keep this approach and assign each layer to a colour from a list (red, blue, green, cyan, magenta and yellow), repeating them if necessary, and then using simple sum to merge all the layers. It may introduce ambiguities, but our results show that this is not relevant at this level of analysis.

6.3.1 Similarity Measure

Since the visualization task, presented in the previous chapter, was focused on showing dependencies between tracks, less attention was paid to the choice of similarity measure. With the music segmentation goal in mind, it becomes an important factor that has to be investigated. The question is now, which similarity measure gives a good separation of different sections (for boundary detection tasks), displays uniform

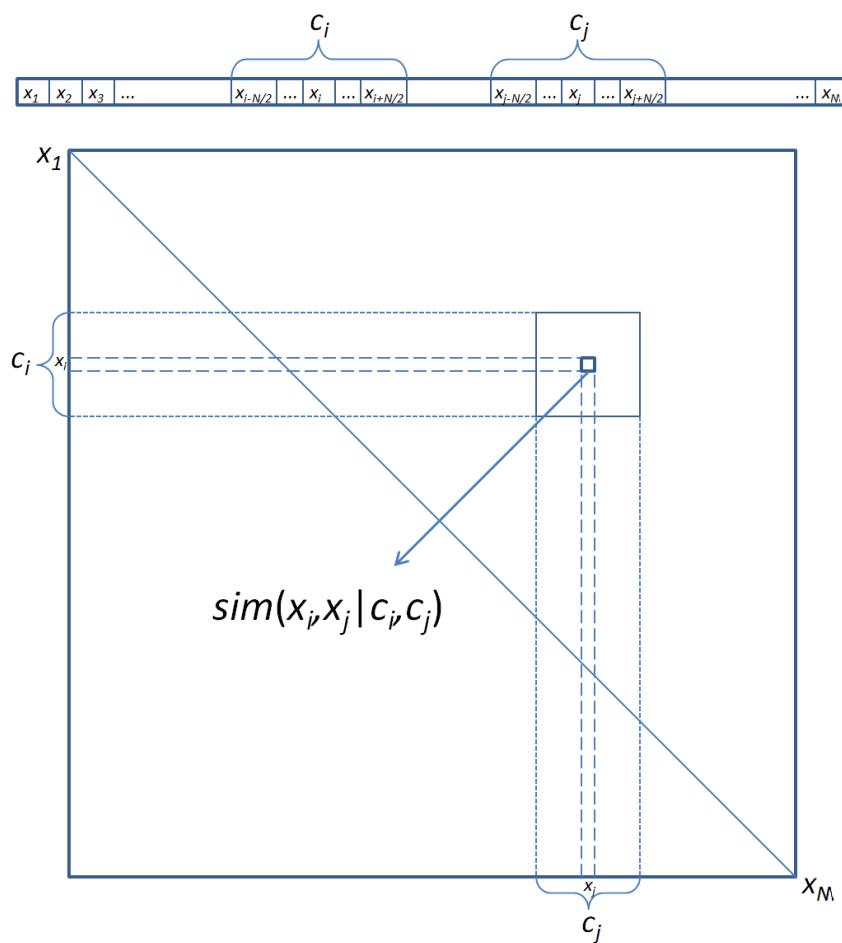


Figure 6.1: The method of obtaining similarity matrices from symbolically represented music pieces. The similarity value at points at x_i and x_j results from similarity score obtained for contexts c_i and c_j , of length N .

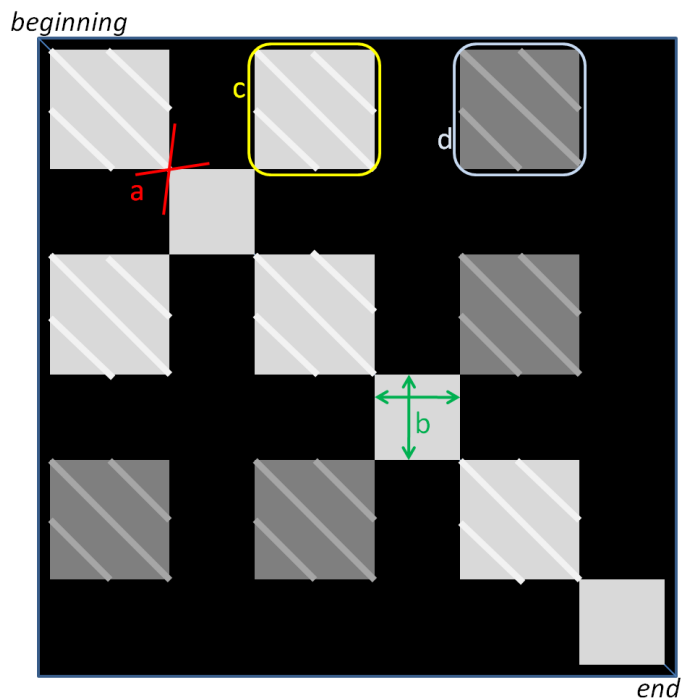


Figure 6.2: Good similarity measure should allow to detect section boundaries (a), homogeneous regions (b), and repetitions of identical (c) and similar (d) sections

patterns or texture between section boundaries (homogeneity detection) and identifies repetition of similar sections by highlighting areas off the diagonal (repetition and section dependencies detection). Figure 6.2 illustrates how a good similarity measure should perform with regard to those requirements.

Among all the methods used in the literature to measure similarity between texts, one can distinguish two main groups — order-aware and order-free. The first group keeps the order of tokens (or features) and a small shift of context may substantially change the similarity value. Generally, theme melodies are usually quite complicated in contrast to accompaniment, i.e., the same notes or relations between notes do not repeat in close proximities or if they do, it is done purposely by the composer. As a result, using order-aware similarity measures makes similar patterns appear as distinct linear patterns on similarity matrices. On the other hand, order-free methods should be able to mask that and blend certain sections to form uniform blocks.

Dealing with features from a given context in a completely order-independent way acts against one of the key benefits of using symbolic representations: preserving dependencies between notes. To compensate for this, the algorithms will be analyzed with note n -grams, which are created based on each n consecutive uni-grams, in addition to features that are derived from single notes.

The uni-gram feature set \mathcal{U}^1 for a context \mathbf{c}_i is given as:

$$x_k \in \mathcal{U}_{\mathbf{c}_i}^1 \iff \exists_v : c_{i_v} = x_k \quad (6.2)$$

where $\mathcal{U}_{\mathbf{c}_i}^1$ is simply a set of all elements contained in otherwise ordered sequence \mathbf{c}_i . Likewise, n -gram feature set \mathcal{U}^n for a context \mathbf{c}_i could be denoted as:

$$(x_k, \dots, x_{k+n-1}) \in \mathcal{U}_{\mathbf{c}_i}^n \iff \exists_v : (c_{i_v}, \dots, c_{i_v+n-1}) = (x_k, \dots, x_{k+n-1}) \quad (6.3)$$

so $\mathcal{U}_{\mathbf{c}_i}^n$ contains all the subsequences of consecutive elements within context \mathbf{c}_i .

In the following paragraphs various text-based distance measures used in our experiments will be introduced. All of them calculate distance based on the contexts of the notes for which the similarity is measured, but the resulting similarity will affect only a very precise point (with (x_i, x_j) coordinates) on the matrix. The similarity formulae given below typically apply to $\mathcal{U}_{\mathbf{c}_i}^1$, but they can be adjusted to any n -gram length feature sets, so for simplicity of notation, we would use $\mathcal{U}_{\mathbf{c}_i}$ as a notation for a general feature set of context \mathbf{c}_i with some arbitrary n chosen a priori.

Hamming distance. This distance measure treats both contexts as vectors and the similarity is equal to the number of the same components in both vectors (the order of features is preserved). It can be denoted as:

$$\text{sim}(x_i, x_j | \mathbf{c}_i, \mathbf{c}_j) = |\{c_{i_k} | c_{i_k} = c_{j_k}\}| \quad (6.4)$$

This measure is fully order dependent and a slight misalignment of the elements of vectors could result in no similarity. This approach has been used in the previous work [209] that was focused solely on the visual part of the problem.

Levenshtein edit distance. Edit distances are well known text similarity measures to calculate the distance between two arbitrary texts. The result reflects the number and type of edition operations that transform from one text to another. Levenshtein edit distance, employs three operations: deletion, insertion and substitution, giving them equal weights.

$$sim(x_i, x_j | \mathbf{c}_i, \mathbf{c}_j) = lev_{\mathbf{c}_i, \mathbf{c}_j}(|\mathbf{c}_i|, |\mathbf{c}_j|) \quad \text{where} \quad (6.5)$$

$$lev_{\mathbf{c}_i, \mathbf{c}_j}(m, n) = \begin{cases} m + n & \text{if } mn = 0 \\ \min(1 + lev_{\mathbf{c}_i, \mathbf{c}_j}(m - 1, n), \\ \quad 1 + lev_{\mathbf{c}_i, \mathbf{c}_j}(m, n - 1), \\ \quad 0 + lev_{\mathbf{c}_i, \mathbf{c}_j}(m - 1, n - 1)) & \text{if } mn > 0 \wedge c_{i_{m-1}} = c_{j_{n-1}} \\ \min(1 + lev_{\mathbf{c}_i, \mathbf{c}_j}(m - 1, n), \\ \quad 1 + lev_{\mathbf{c}_i, \mathbf{c}_j}(m, n - 1), \\ \quad 1 + lev_{\mathbf{c}_i, \mathbf{c}_j}(m - 1, n - 1)) & \text{if } mn > 0 \wedge c_{i_{m-1}} \neq c_{j_{n-1}} \end{cases}$$

The algorithm finds the optimal sequence of transitions from one text to another in $O(n^2)$ time where $n = \|\mathbf{c}\|$, so in total it gives $O(N^2n^2)$ where N is the number of notes in the piece. Assuming that n is constant, this does not change the general complexity of the algorithm, however it is still considerably slower than other measures, which perform in at most $O(n \log n)$ time.

Edit distance algorithms are order-aware, albeit the ability to shift the entire context (by a deletion at one end and an insertion at the other) makes them able to accommodate for some shifts in the contexts. This causes the maximal difference of similarity between two neighbouring points on the matrix less than or equal to 2 and hence leads to smoother results.

Cosine similarity. This measure belongs to the group of order-free methods (also dubbed bag-of-words approaches) and is commonly used to compare texts in information retrieval. It is expressed as the cosine value of an angle between two document vectors where each vector contains values of frequencies of each term in

the documents in question. Formally, cosine similarity it is a dot product of two term vectors:

$$sim(x_i, x_j | \mathbf{c}_i, \mathbf{c}_j) = \frac{\sum_{w \in \mathcal{U}_{\mathbf{c}_i} \cup \mathcal{U}_{\mathbf{c}_j}} f_{i_w} f_{j_w}}{\sqrt{\sum_{w \in \mathcal{U}_{\mathbf{c}_i}} f_{i_w}^2} \sqrt{\sum_{w \in \mathcal{U}_{\mathbf{c}_j}} f_{j_w}^2}} \quad (6.6)$$

where f_{i_w} denotes the frequency, or the number of occurrences of a certain term, or n-gram w in context c_i , that is:

$$f_{i_w} = |\{k \mid w = c_{i_k}\}|. \quad (6.7)$$

This measure should perform well with respect to hiding irrelevant details, unveiling the general structure of the piece.

Dice's index. Dice's index [32] can be seen as a special version of cosine similarity with binary weights applied to the feature set, i.e., each feature, or n-gram weight is either 0 or 1 depending on its existence in the input vectors. Since the number of times a feature occurs in the input string does not matter, the measure should focus on more complicated passages, that is, those that contain more distinct features. This is a slight modification of the original formula that normalizes the result by the size of context, not feature vector:

$$sim(x_i, x_j | \mathbf{c}_i, \mathbf{c}_j) = \frac{2 \|\mathcal{U}_{\mathbf{c}_i} \cap \mathcal{U}_{\mathbf{c}_j}\|}{|\mathbf{c}_i| \cdot |\mathbf{c}_j|} \quad (6.8)$$

Jaccard index. This is another variation on binary-weight cosine similarity where the result is normalized with respect to the total number of features that occur in either vectors [86]. This should give more focus to background repetitive patterns:

$$sim(x_i, x_j | \mathbf{c}_i, \mathbf{c}_j) = \frac{\|\mathcal{U}_{\mathbf{c}_i} \cap \mathcal{U}_{\mathbf{c}_j}\|}{\|\mathcal{U}_{\mathbf{c}_i} \cup \mathcal{U}_{\mathbf{c}_j}\|} \quad (6.9)$$

CNG measure. This method calculates similarity over n-gram profiles that consist of frequency counts as defined in Equation 6.7. The similarity measure as defined below is a linear transformation of the original distance measure by Keselj et al. [93], which was proven to be useful in distinguishing between authors of texts

as well as composers in music domain [215], making it an interesting candidate for application to this task. The formula for the similarity measure is given as follows:

$$\text{sim}(x_i, x_j | \mathbf{c}_i, \mathbf{c}_j) = \sum_{w \in \mathcal{U}_{\mathbf{c}_i} \cup \mathcal{U}_{\mathbf{c}_j}} \left(1 - \left(\frac{f_{i_w} - f_{j_w}}{f_{i_w} + f_{j_w}} \right)^2 \right) \quad (6.10)$$

The goal of this analysis is to assess suitability of certain text-based similarity measures for the structure analysis task. A good measure should perform well for both classical and popular music, so one does not have to design separate approaches to both areas. Since popular music has usually simple structure and demonstrates less complicated internal relations between fragments, as an initial benchmark we decided to take first 32 bars of Prelude C# major from “Das Wohltemperierte Klavier I”. This excerpt contains repeating fragments and pattern that migrate between voices, to different registers, and which are played in parallel keys. A good similarity measure should capture, what is similar, and discard, or at least diminish the influence of the differences.

Figure 6.3 contains results of analysis of this excerpt. One can notice that simple Hamming distance measure is not capable of capturing structure very well, regardless of the feature extraction method it employs. Figure 6.3a uses exact notes pitch and duration matching, Figure 6.3b takes pitch information modulo 12, so in a sense employs the equivalent of chroma features used in audio-based applications, and Figure 6.3c and d employ the measure where relative pitch and duration are being used. One can notice that using relative features helps in extracting patterns but not structure and that Levenshtein edit distance better averages little fluctuations in the patterns and tend not to cross section borders, but that is not sufficient to reveal the full structure in this particular example.

Figure 6.4 displays results for order-free functions. One can observe that they behave much better for larger context sizes and with features based on intervals and relative durations rather than pitch classes. All of the proposed bag-of-words algorithms perform well in determining the structure in this case, so in order to find which of them performs better than others, one would have to look at a bigger

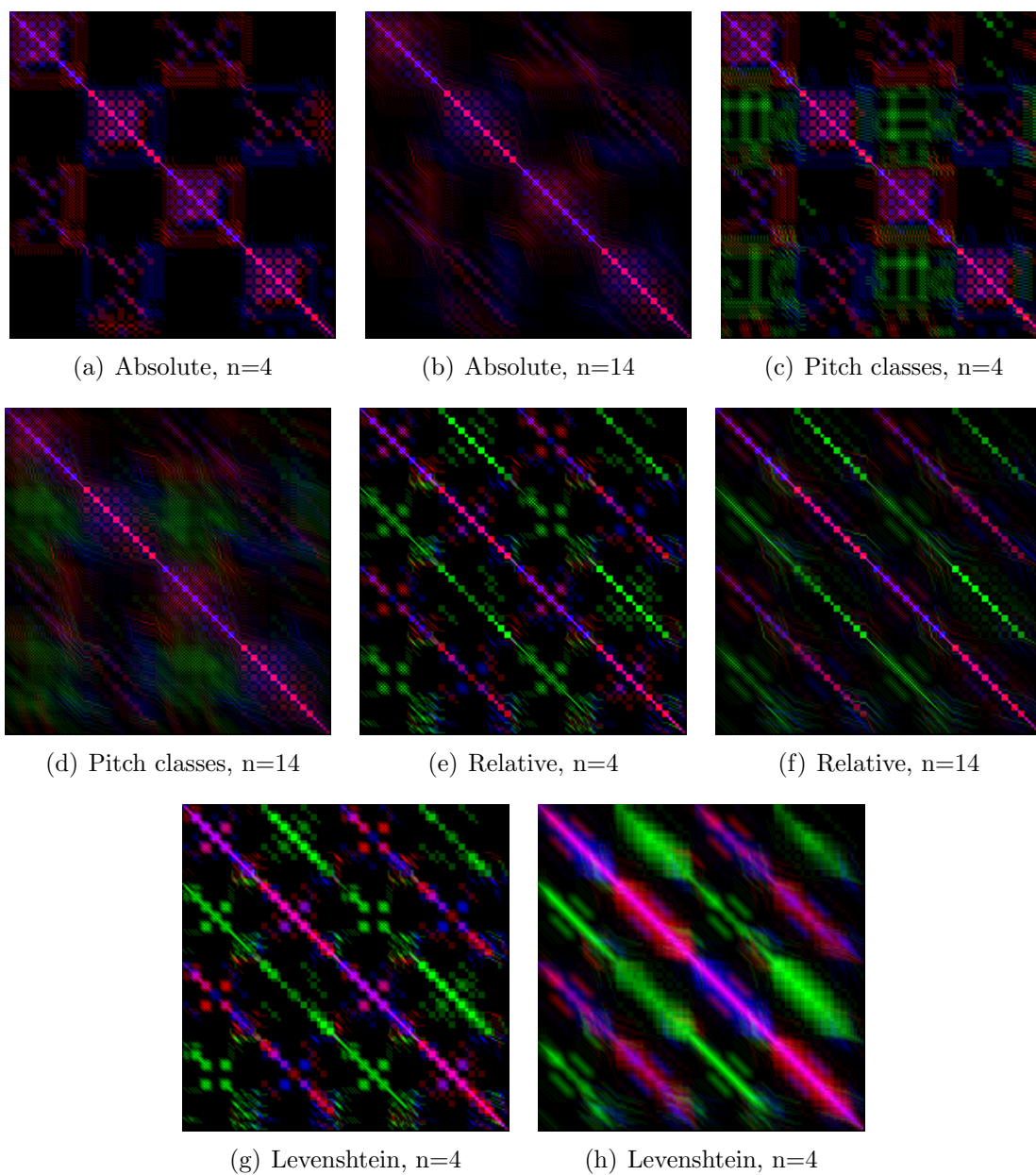


Figure 6.3: Similarity matrices for test input using Hamming distance with absolute, pitch classes and relative methods of feature extraction (a–f) and Levenshtein edit distance (g, h)

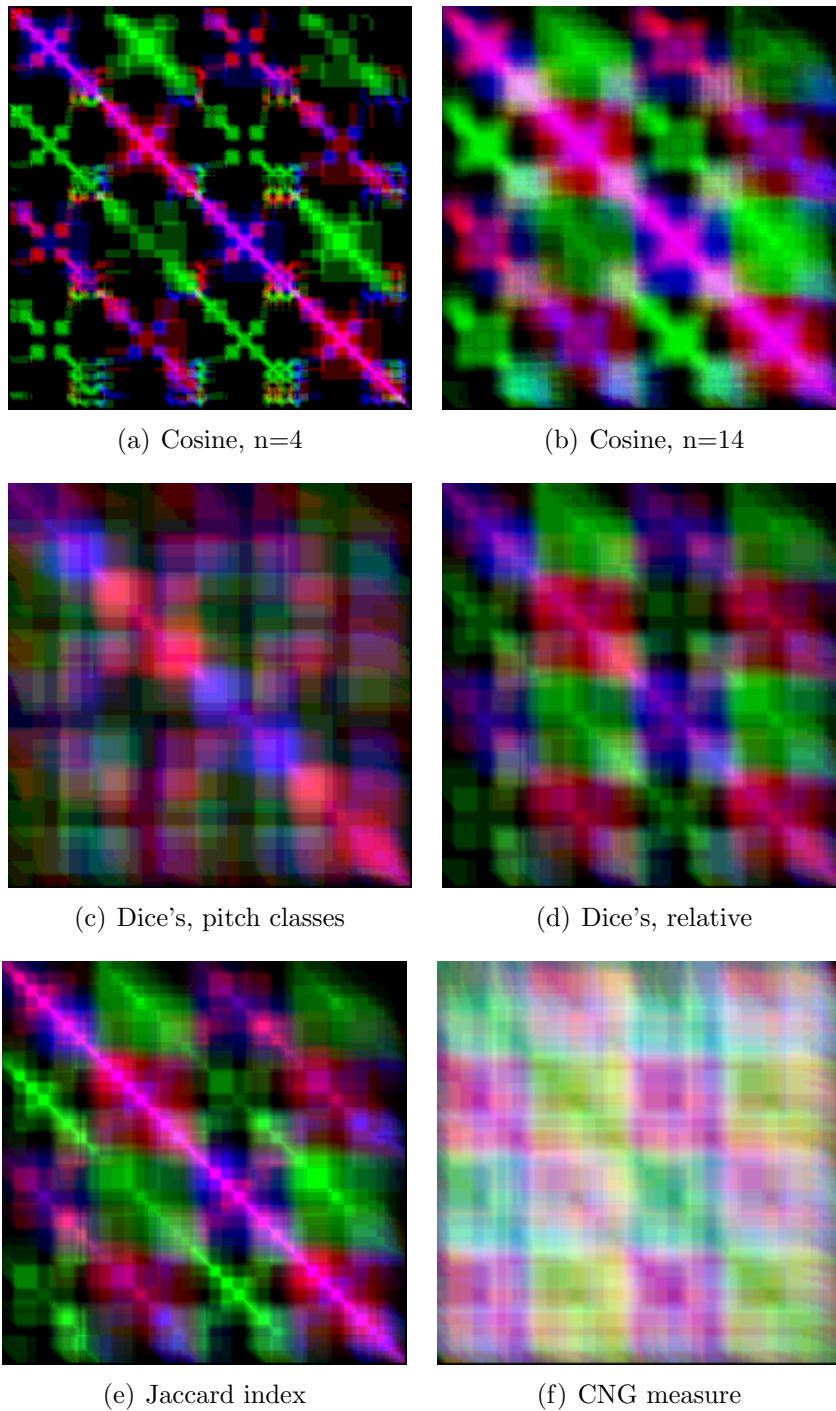


Figure 6.4: Similarity matrices for test input using bag-of-words approaches: Cosine similarity (a–b); with $n=14$: Dice's index with pitch classes and relative features (c–d), Jaccard index (e) and CNG measure (f)

example. Figure 6.5 contains results of the analysis of the entire prelude. The best performer is cosine similarity measure with bi-gram features where all the different and similar sections are clearly visible. It shows a clear distinction of subsections within sections and indicate sections that are not similar at all by keeping the relevant areas on the matrix dark. This is not the case with the CNG measure (Figure 6.5f) where almost entire similarity matrix indicates that all sections are related to each other, albeit it clearly indicates, with specific colour transitions, where each section begins and starts, which indicates its potential with novelty detection techniques.

The segmentation algorithm works for popular music as well. Figure 6.6 contains analyses of the same measures as in Figure 6.5, but for the Beatles song, “Magical Mystery Tour”. It has a tripartite structure with an instrumental section in the middle and a long coda. One can draw the same kind of conclusions about similarity measures in question as for Bach’s Prelude, despite a different nature of those two pieces.

To conclude, for all the introduced similarity functions, namely Hamming distance (order aware) used in previous work [209], Levenshtein edit distance (order-aware) and cosine similarity, Dice index, Jaccard index and CNG similarity measure (all order-free), we have made a number of qualitative comparisons between them. We observed that some perform well in identifying (visualizing) repetitive patterns (Levenshtein) while some focused mainly on identifying homogeneous parts (CNG measure, see Figure 6.5). The similarity function we have chosen for future analysis was cosine similarity as it performed well in distinguishing similar and different sections as well as identifying repetitive patterns.

6.3.2 Similarity Matrices Adjustments for Structural Analysis

In order to perform effective structural analysis, we had to make a number of adjustments to the original visualization technique. First of all — having multiple layers, one for each pair of voices, increases the ability to display more information on a single image but may create ambiguity in further analysis. Moreover, keeping layers

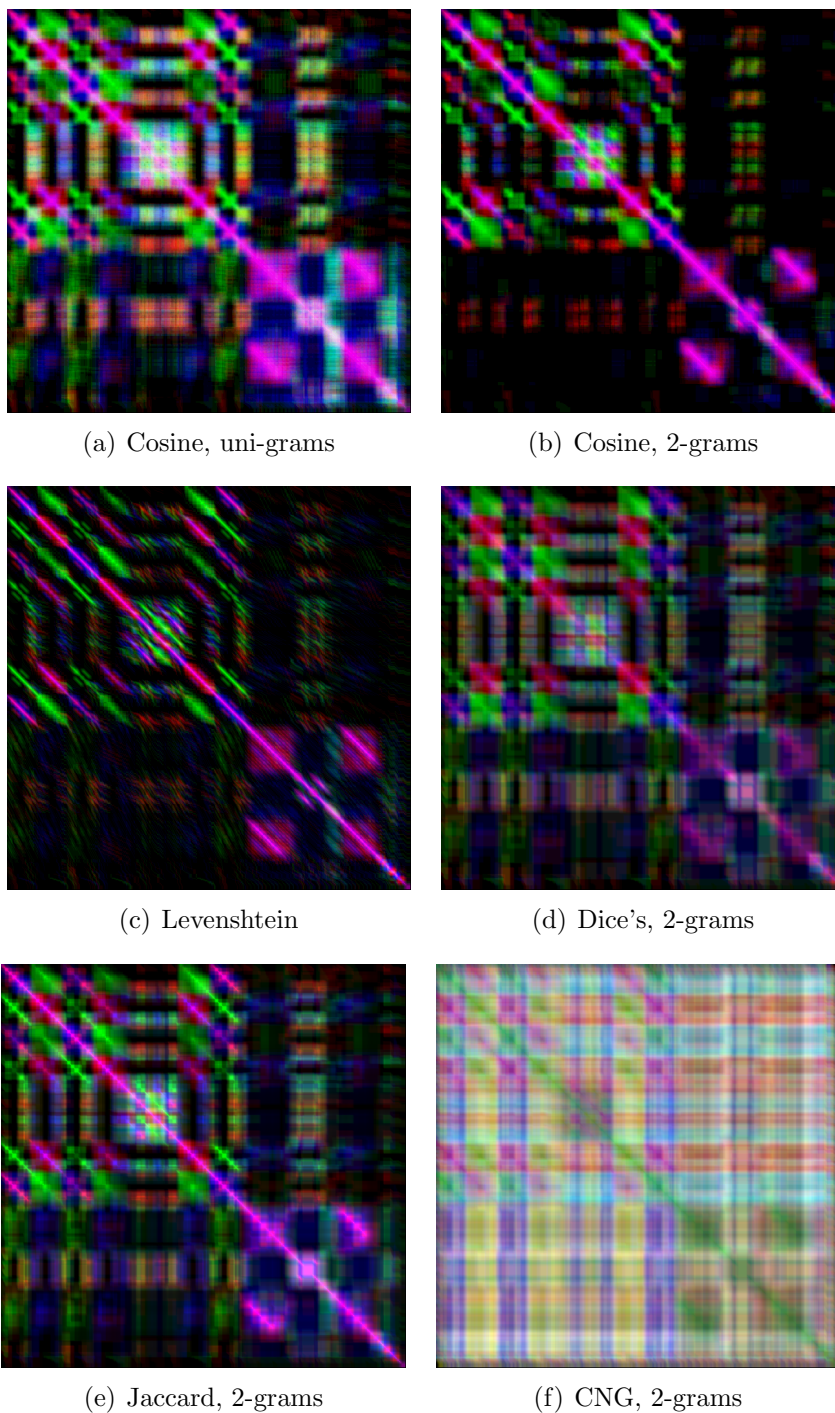


Figure 6.5: Similarity matrices for Bach Prelude C# major, all $n=14$: Cosine similarity (a–b), Levenshtein (c), and Dice's (d), Jaccard (e) and CNG (f) with 2-grams.

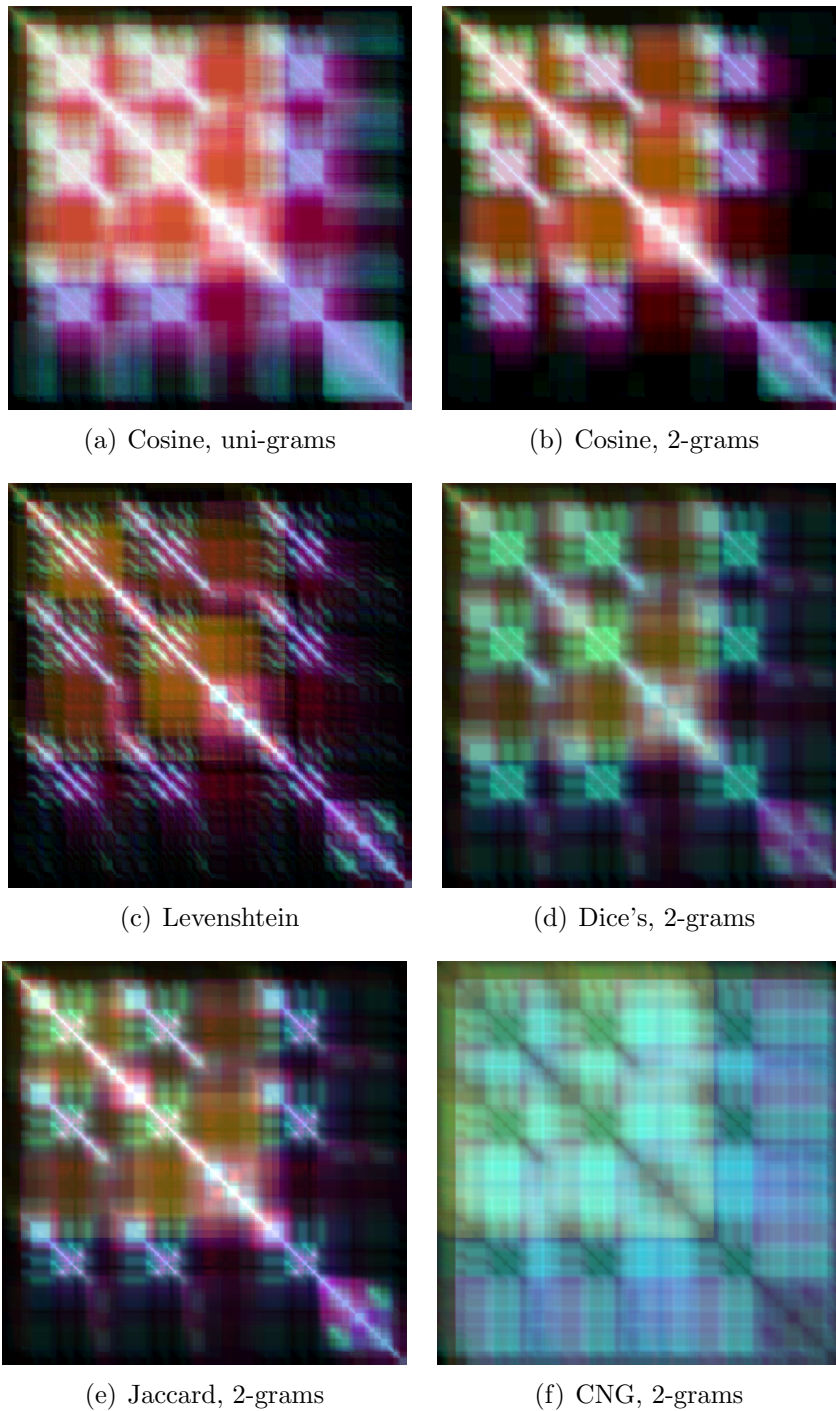


Figure 6.6: Similarity matrices for the Beatles' "The Magical Mystery Tour", all $n=14$: Cosine similarity (a–b), Levenshtein (c), and Dice's (d), Jaccard (e) and CNG (f) with 2-grams.

separate results in asymmetric image which would lead to unnecessary complications. Therefore we have decided to merge layers such that similarity at each point reflects similarity between n-gram sets from all currently active tracks. As a result of that, obtained similarity matrices are monochrome and symmetric along the diagonal, so one can focus only on one side of the similarity matrix for analysis purposes.

Some adjustments have been made to the timing parameters. Previously similarity has been defined between two notes (or in fact, two uni-grams) each with the same context size of N notes. As notes are not well defined for audio-based structure segmentation systems, it's common to use actual durations (in seconds) to define temporal parameters of the analysis. Isaacson [85] demonstrated how greatly music performers tend to deviate from the set metric time into more irregular patterns, pointing the importance of deciding, whether to use the real time flow, or metric time of a music score. Since other approaches that we will be using for comparison use real audio time, we have decided, for compatibility, to switch to it as well, using seconds for analysis resolution and context size. As a result — to create a vector at each point (which is then compared against other vectors), we have to define a window size, or resolution (r), and offset (o) in seconds. First defines a window that contains all notes played during a given time r . A stream of notes for each track is then expanded by N notes, where N is calculated as the average number of notes that fall into o seconds. It is fixed for all layers and allows for more balanced measuring of cosine similarity between vectors (i.e., the size of each vector is at least $2 * N * \#tracks$ plus all the notes that fall into the r gap)(see Figure 6.7).

We usually refer to each step with its contexts as a music phrase, which length is proportional to N . Similarly, the choice of n-gram length, n , reflects the size of a music word. According to our previous analysis [213], it turned that a choice of $n = 2$ or $n = 3$ gives good results for determining symbolic music similarity, so those values will be our preferred n-gram lengths.

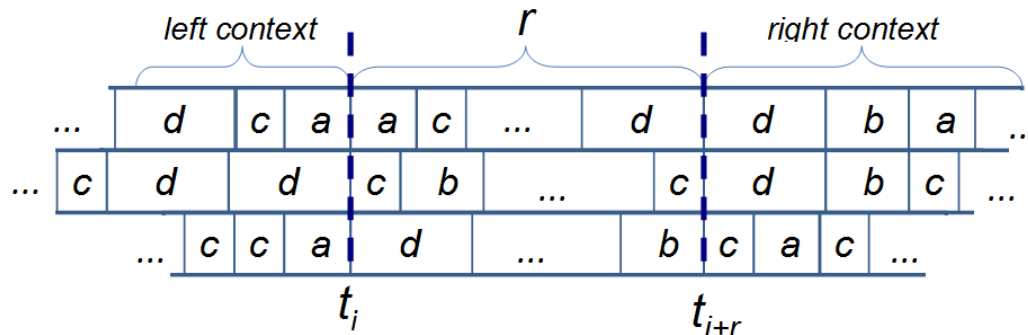


Figure 6.7: An example showing how layers are joined to form one vector of features corresponding to t_i . Here the context size of $N = 3$ notes is used. All of the features (a's, b's, c's and d's) on the drawing conform the feature vector at t_i .

6.3.3 Novelty Detection Function

Novelty detection is one of the techniques used frequently in music structure analysis task. It can be used as it is, as a boundary detection algorithm, but we will use it as a helper function for the proposed modification to Dijkstra path-finding algorithm. Novelty function was originally proposed by Foote [52], where he was convolving the diagonal of a self-similarity matrix with a checkerboard kernel and this is a common technique to determine novelty in other structure analysis systems. However, this solution acts locally, i.e., only points in the local context to the point we are currently evaluating count towards the novelty value. Since music has its repetitive nature, a strong boundary can be reinforced by another, previous (directly above the analyzed point) or subsequent (directly below the analyzed point) occurrence of a similar section boundary. Therefore it is reasonable to compute novelty function using not only local context near the diagonal, but through the entire similarity matrix. However, it looks probable that calculating local novelty function on a second order similarity matrix as proposed by Peeters [149] would lead to similar results. At each point of time t_x , given self-similarity matrix $\mathbf{S} = [a_{i,j}]_{M \times M}$, novelty $nov(x)$ is defined as:

$$nov(x) = \sum_{i=0..M} \left| \sum_{j=x-N..x} a_{i,j} - \sum_{j=x..x+N} a_{i,j} \right| \quad (6.11)$$

Novelty is then normalized to the range of values $[0, 1]$. Peaks in value indicate rapid change of music structure (and a possible boundary).

6.3.4 Modified Dijkstra Algorithm

Self-similarity matrix is organized in a way that all the points above the diagonal (upper triangle) refer to the similarity of the notes on the diagonal to the notes before the current one, i.e., the past. Likewise, points below the diagonal (lower triangle) refer to the future, and points on the diagonal — to the present. To detect repetitive patterns in similarity matrices one can find a path from the beginning of the piece to its end, such that one can only move on the diagonal, or jump and move on the upper triangle on the patterns that occurred in the past. The shortest path, assuming it is less expensive to travel through areas of higher similarity, contains the analysis of the patterns. We allow the pathfinder to make jumps to the areas of higher similarity and those jumps will indicate section boundaries. Only the following moves within the upper triangle of a similarity matrix $\mathbf{S} = [a_{i,j}]_{M \times M}$ are allowed:

- $x_{i,j} \rightarrow x_{i+1,j+1}$ with a cost of $1 - a_{i+1,j+1}$ (when a new pattern is in the same tempo as the original),
- $x_{i,j} \rightarrow x_{i+1,j}$ with a cost of $1 + (1 - a_{i+1,j})$ (when a new pattern is slower than the original),
- $x_{i,j} \rightarrow x_{i,j+1}$ with a cost of $1 + (1 - a_{i,j+1})$ (when a new pattern is faster than the original), or
- a jump $x_{i,j} \rightarrow x_{i,k}$ where a jump is at least a phrase long, i.e., $|j - k| > N$ and the cost of a jump is the sum of jumping from j : $N(1 - nov(i))$, jumping to k : $N(1 - nov(i))$ and a regular cost of moving to k : $1 - a_{i,k}$.

The cost of jumps ‘from’ and ‘to’ can be waived based on the history of the previous jumps. Theoretically, a jump from (i, j) to (i, k) indicate section boundaries at i , j and k so if a walker already made decisions of defining those boundaries, it should be free for him to make this jump again.

The algorithm stops when the right side of the matrix is reached, i.e., the current shortest path ends in (M, j) for any j . Of course, based on those assumptions, the shortest path would be directly from $(0, 0)$ to (M, M) through the diagonal (similarity on the diagonal is always 1); therefore the diagonal needs to be pruned to discard this trivial solution. We propose a non-parametric solution where, while calculating the dot product for cosine similarity between two points, we ignore n-grams in both vectors that correspond to exactly the same notes from the piece. As a result of that, as we get closer to the diagonal, more and more n-grams are excluded, resulting in lower similarity, to the points on the diagonal, where all n-grams from both vectors correspond to the same notes and similarity is 0. Since going back to the diagonal should always be allowed (when there is no other patterns that occurred in the past for some i) the similarity on the diagonal should be assigned some value (parameter d) to allow diagonal passes. Typically, values 0.2–0.6 work fine. Lower values of d allow for more flexible dealing with variations of a theme and higher d impose more strict theme similarity checks.

6.3.5 Automatic Structure Inference

Given the shortest path through the matrix, one can derive the structure of the piece. Since only a few borders are found at the first place (the algorithm finds only all necessary borders to travel through the graph), all others have to be derived from the existing ones. If at some point (i, j) the path crosses one of the existing borders, either vertical i or horizontal j , the other one should be added to the border set. Since adding a border can create a new crossing point in a different spot on the path — the process needs to be repeated until there are no new borders found.

Each group (bounded by two consecutive borders) is assigned a group ID, starting from the first group defined between the beginning of the piece and the first detected border. For each next group, if there is no path passing above the current segment, a new ID is assigned to it; otherwise it is assigned an ID of a group, the path points to

There is other problem of doing research in the symbolic music domain — the lack of well annotated datasets of aligned audio and symbolic music. It is understandable as creation of such datasets require enormous amount of expert manual work, unlike in some other areas where community collaboration and social media can be used for data gathering.

We have applied our algorithm to the Beatles song dataset, consisting of 209 popular music pieces, achieving very good segmentation and song interpretation results. With this dataset, we have identified one issue with the algorithm — when some area is dominated with ostinato of quickly repeating short patterns or notes (which happen quite frequently in popular music, but not as much in classical music), the algorithm starts to multiply irrelevant borders that trigger creation of even more borders which at the end creates groups of packed borders appearing close to each other. Thankfully this problem is quite easy to detect and fixed by merging those boundaries to form a separate section. To deal with it, a post-processing rule has to be set, that whenever a short section, typically representing the shortest repeating fragment in the solution, repeats many times (e.g., at least 4 times) in a row, this segment is marked as ostinato and is merged across the entire piece with other segments with the same label, that appear next to each other.

Based on SALAMI (Smith et. al [171]) dataset, we have extracted a subset of classical music pieces (which was a part of IA (Internet Archive) corpus, prepared and annotated by Smith et al. [170]), along with the results of 5 algorithms operating on audio data supplied with SALAMI-IA corpus:

- `barrington` (Barrington et al. [5]),
- `echonest` (Jehan and DesRoches [89]),
- `levy` (Levy and Sandler [112]),
- `peiszer` (Peiszer [150]), and
- `xavier` (Mestres [131]).

We have manually searched for MIDI files of well-sequenced counterparts of files in the SALAMI-IA corpus and we have managed to find 11 out of 15, which are:

1. **Suite in D** for orchestra by George Handel,
2. **Nutcracker Suite no. 2, “Marche”** for orchestra by Peter Tchaikovsky,
3. **Symphony no. 8, op. 93, third movement** for orchestra by Ludvig van Beethoven,
4. **Symphony no. 40 in G minor, K. 550, third movement** for orchestra by Wolfgang Amadeus Mozart,
5. **Piano Sonata no. 8 op. 13 (“Pathétique”), second movement** for piano by Ludvig van Beethoven,
6. **Sonata no. 14 (“Moonlight”), op. 27 no. 2, second movement** for piano by Ludvig van Beethoven,
7. **L’arlésienne: Suite no. 1 — 2nd movement, Menuet** for orchestra by Georges Bizet (marked in the SALAMI-IA corpus by Smith et al. [170] as Unknown),
8. **Gigue en Rondeau** for piano by Jean-Philippe Rameau,
9. **Symphony no. 1, third movement** for orchestra by Robert Schumann,
10. **Minuet in G major, op. 14 no. 1** for piano by Ignacy Jan Paderewski, and
11. **Minuet in G minor** for piano by Christian Petzold.

After that, MIDI files were manually aligned with corresponding audio files to perform meaningful comparison of the results of those algorithms. We have used the best parameters reported by Smith for all five algorithms. For our analysis we have used heuristically optimal parameter set ($r = 0.5s$, $o = 2.5s$, $n = 3$, $d = 0.3$), estimated based on our analysis of Beatles corpus. The SALAMI-IA corpus is supplied

with two sets of annotations, fine and coarse, to accommodate ambiguous nature of determining music structure. We evaluated the algorithms against both of them. Smith allowed an algorithm to miss a section border by 0.5s or 3s. We have found 3s too large as with fine annotations, it was common that error margins have spanned most of the length of a short and dense piece. In our evaluations we have used allowed error margins of 1s. Evaluations of all the pieces mentioned above, along with manual alignment can be reviewed on Figure 6.10. One can observe, that some MIDI pieces do not align exactly with corresponding audio files, i.e., piece #4 has one repeated section missing in MIDI, in #8 has some content entirely skipped comparing to its audio counterpart, and some pieces have missing silence periods at the front and at the back of a piece, so adjustments have been made to the evaluation data to reflect those changes. Same colour has been applied to the parts marked with the same label for a given segmentation. However, **echonest** and **xavier** do not label sections, and labelling is not part of the evaluation, so colours are provided here just for reference.

Each algorithm, for each piece and each gold standard, has been given a score, based on the number of correctly assigned borders. Two basic measures have been applied: **precision**, as the ratio between the number of correctly assigned (matching) borders and the total number of borders returned by the algorithm, and **recall**, as the ratio of correctly assigned (matching) borders to the total number of borders defined by a gold standard. To balance precision and recall, the **F-measure** is frequently used, which is their geometric average. All the partial results are then averaged for each algorithm and gold standard for cumulative precision, recall and F-measure (see Figure 6.11). We have found that our algorithm performed much better than any other algorithm operating on audio data. In many individual cases we were able to score either $P = 1$ or $R = 1$, although it is almost impossible to score $F = 1$ due to ambiguous nature of music annotation. Our algorithm scored for coarse segmentation $F = 0.75$ versus the second next, **xavier** with $F = 0.42$. Similar result was achieved for fine segmentation where our algorithm achieved $F = 0.67$ versus second best, **xavier**'s $F = 0.40$. To see how far apart was the performance our

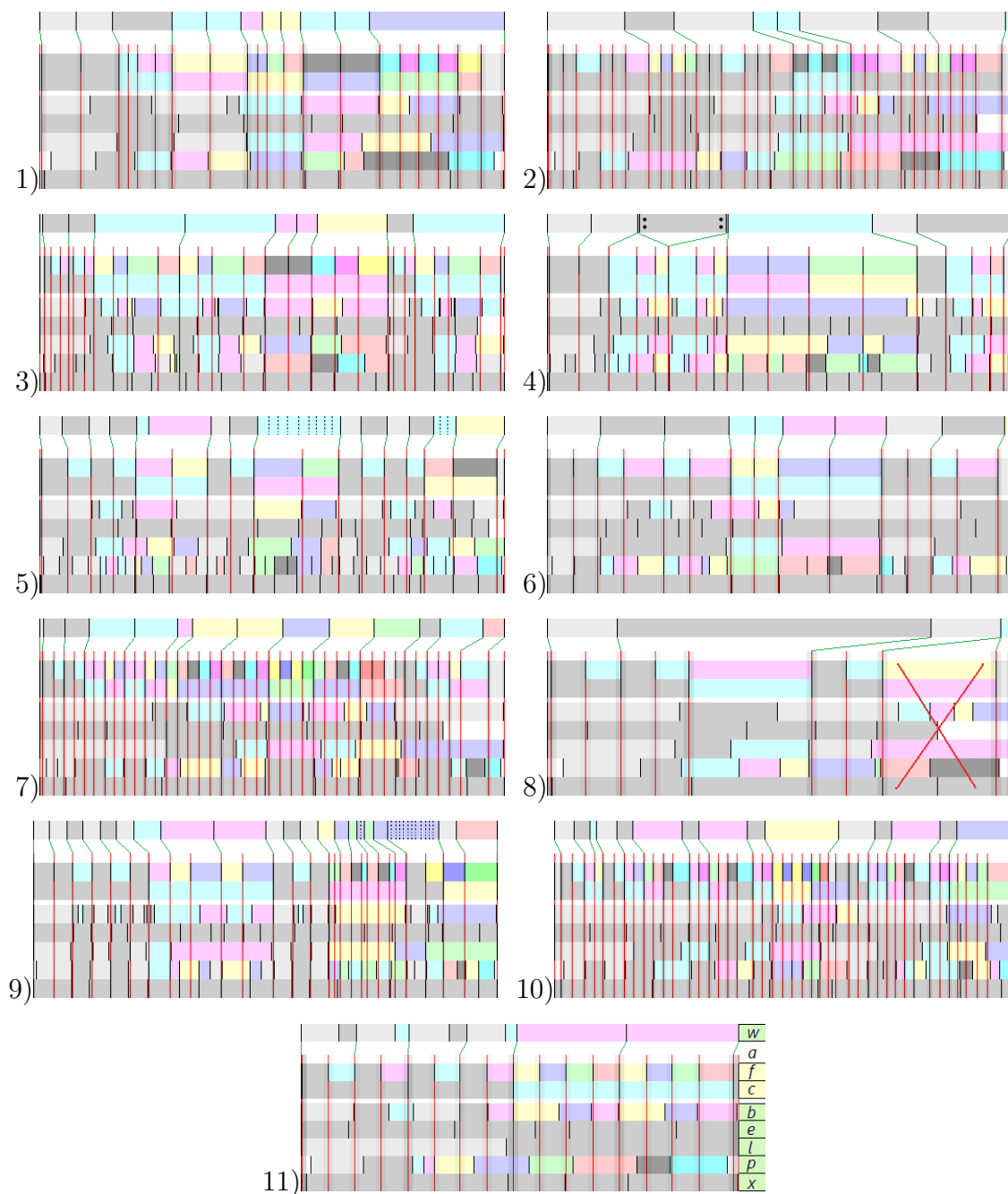
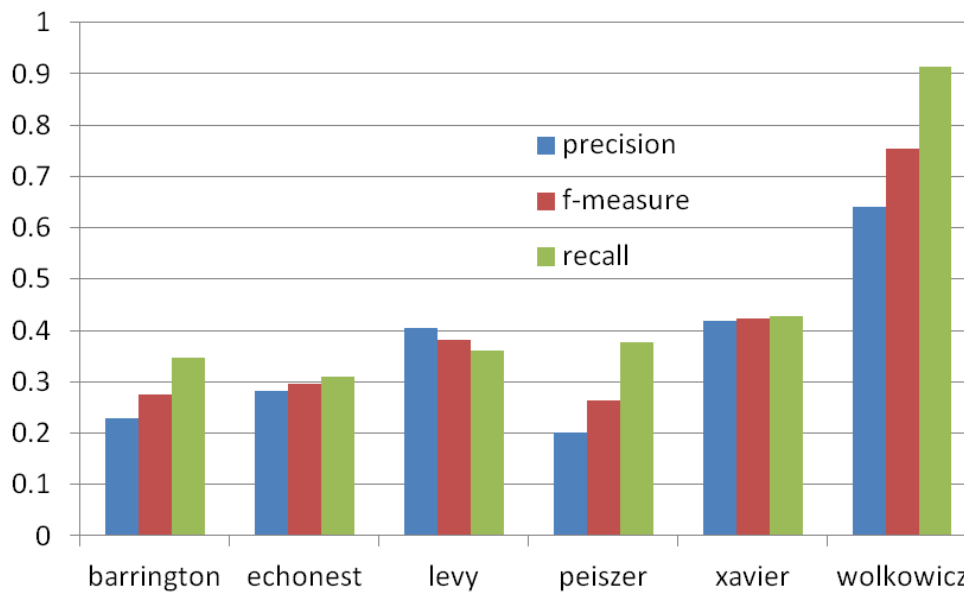


Figure 6.10: Structural segmentation of eleven pieces from IA corpus. Each figure consists of our analysis of MIDI data (**w**), followed by MIDI-audio alignment (**a**), fine and coarse gold standard (**f**, **c**), followed by results obtained from other evaluated algorithms (barrington (**b**), echonest (**e**), levy (**l**), peiszer (**p**), and xavier (**x**)).

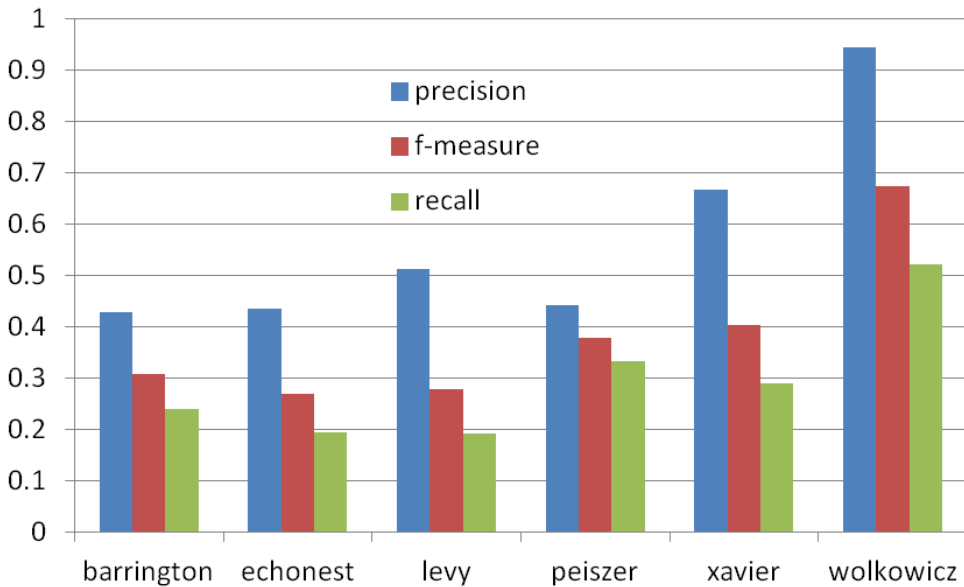
algorithm comparing to other algorithms operating on audio data, Figure 6.12 shows precision-recall graph of the obtained results, with error bars indicating standard errors of the average precision and average recall estimations.

To quantify the significance of the obtained results, paired t-test have been performed on all partial precision, recall and F-measure results. The p values, indicating the probability of the null hypothesis, that there is no difference between corresponding performance results, have been collected in Tables 6.1 and 6.2. Colour indicates where the probabilities of the null hypothesis falls below 5%, so one can say that the difference is statistically significant. Red indicates that algorithm in a given row performs significantly worse than the algorithm in the column, while green means that it performed significantly better. We can see that our approach outperformed all other algorithms consistently for all measures and settings. One can also note a good performance of **xavier** for both segmentations, **levy** for coarse and **peiszer** for fine gold standard.

Although our algorithm performed significantly better than other, based on audio data, there is still a room for improvement. Primarily, we still do not know how much of the success of our system has to be attributed to a different data type used as input, and how much to the actual performance of this algorithm design. Aside from that, we have identified several potential issues. For some classical music pieces, where the structure is defined in a very complex way, that is, it is not demonstrated by changes in rhythmic, melodic, harmonic or phrasing aspects of the piece, the algorithm will not be able find any clear structure. This applies, for example, to fugues. The quality of the results depends also on the quality of the underlying MIDI file. If the file is sequenced in a messy, unorganized way, the resulting structure may be misleading or wrong. With a large number of effect tracks and small number of the actual music tracks, the structure driven by these effect tracks may overload quickly the real structure that is derived from melodic tracks. Our last issue affects mainly popular music, however this problem is actually relatively easy to overcome by detecting those tracks and removing them in preprocessing.

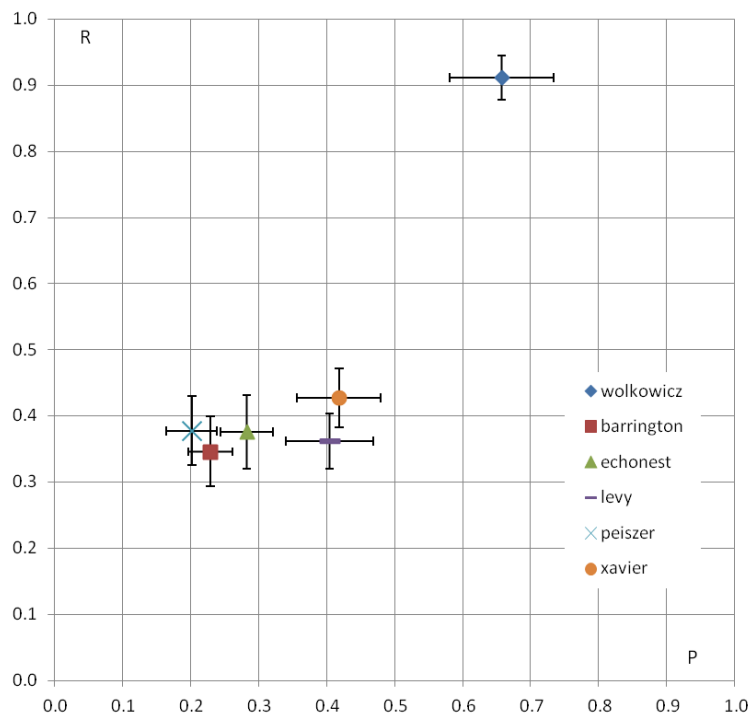


(a) Coarse structure annotations

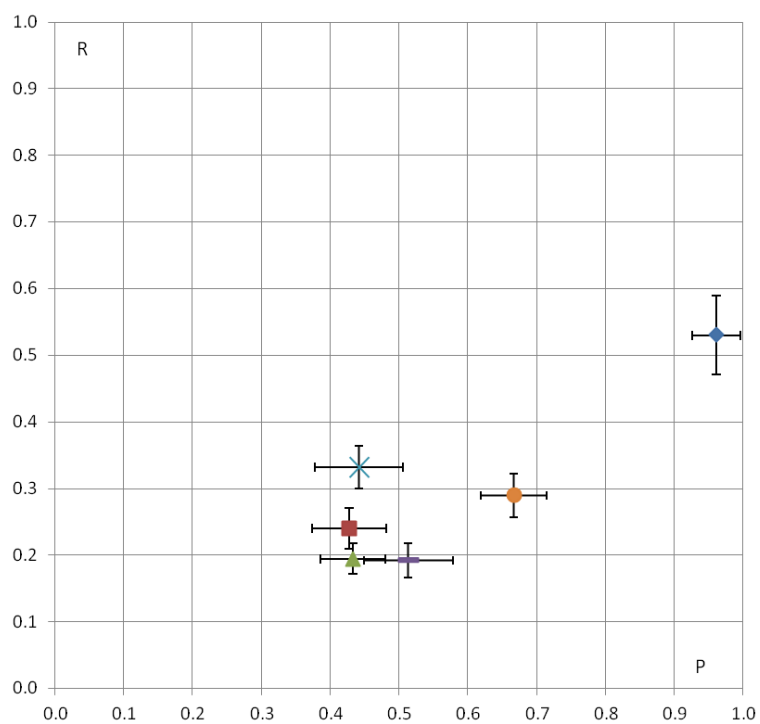


(b) Fine structure annotations

Figure 6.11: Comparison of Precision, Recall and F-measure results on a subset of classical IA Corpus of a set of audio driven segmentation algorithms against the proposed algorithm operating on symbolic data.



(a) Coarse gold standard



(b) Fine gold standard

Figure 6.12: Precision — Recall graphs indicating performance of compared algorithms. Error bars indicate errors of mean estimations for precision and recall separately, indicating separation of obtained cumulative results.

Table 6.1: Significance of the obtained results for coarse segmentation. Values represent the probability that the corresponding algorithms have the same performance, assuming t-distribution of the results. Red and green background indicates significant differences in the performance between an algorithm in a row versus the algorithm in the column (with $p < 0.05$). Red indicates significantly worse performance, while green — significantly better.

(a) precision

	wolk	barr	echo	levy	peis	xavi
wolk	0.500	0.000	0.000	0.010	0.000	0.002
barr	0.000	0.500	0.057	0.001	0.252	0.001
echo	0.000	0.058	0.500	0.013	0.020	0.006
levy	0.010	0.001	0.013	0.500	0.006	0.386
peis	0.000	0.252	0.020	0.006	0.500	0.001
xavi	0.002	0.001	0.006	0.386	0.001	0.500

(b) recall

	wolk	barr	echo	levy	peis	xavi
wolk	0.500	0.000	0.000	0.000	0.000	0.000
barr	0.000	0.500	0.258	0.382	0.334	0.098
echo	0.000	0.258	0.500	0.142	0.169	0.030
levy	0.000	0.382	0.142	0.500	0.419	0.099
peis	0.000	0.334	0.169	0.419	0.500	0.223
xavi	0.000	0.098	0.030	0.099	0.223	0.500

(c) F-measure

	wolk	barr	echo	levy	peis	xavi
wolk	0.500	0.000	0.000	0.000	0.000	0.000
barr	0.000	0.500	0.283	0.005	0.432	0.003
echo	0.000	0.283	0.500	0.040	0.253	0.010
levy	0.000	0.005	0.040	0.500	0.055	0.190
peis	0.000	0.432	0.253	0.055	0.500	0.009
xavi	0.000	0.003	0.010	0.190	0.009	0.500

Table 6.2: Significance of the obtained results for fine segmentation. See Table 6.1 for description.

(a) precision

	wolk	barr	echo	levy	peis	xavi
wolk	0.500	0.000	0.000	0.000	0.000	0.000
barr	0.000	0.500	0.445	0.045	0.398	0.001
echo	0.000	0.445	0.500	0.068	0.445	0.000
levy	0.000	0.045	0.068	0.500	0.107	0.002
peis	0.000	0.398	0.445	0.107	0.500	0.001
xavi	0.000	0.001	0.000	0.002	0.001	0.500

(b) recall

	wolk	barr	echo	levy	peis	xavi
wolk	0.500	0.000	0.000	0.000	0.001	0.000
barr	0.000	0.500	0.049	0.109	0.037	0.118
echo	0.000	0.049	0.500	0.453	0.001	0.002
levy	0.000	0.109	0.453	0.500	0.000	0.001
peis	0.001	0.037	0.001	0.000	0.500	0.123
xavi	0.000	0.118	0.002	0.001	0.123	0.500

(c) F-measure

	wolk	barr	echo	levy	peis	xavi
wolk	0.500	0.000	0.000	0.000	0.000	0.000
barr	0.000	0.500	0.167	0.280	0.109	0.032
echo	0.000	0.167	0.500	0.446	0.024	0.001
levy	0.000	0.280	0.446	0.500	0.009	0.001
peis	0.000	0.109	0.024	0.009	0.500	0.268
xavi	0.000	0.032	0.001	0.001	0.268	0.500

6.5 Conclusions

This chapter introduces an approach to the problem of structural analysis with symbolic music. We have found that symbolic representations are capable of delivering better quality information than audio files which results in more thorough analysis as the context, or the dependencies between notes, plays very important role.

Many popular text-based similarity measures were introduced to enhanced symbolic music visualizations for analysis purposes and we have found that most bag-of-word approaches work well in revealing music structure especially if they incorporate a wide note context, which could be achieved only using symbolic representations. We obtained the best results with cosine similarity measure using bi-gram frequencies. We proposed a broad context novelty function, modified Dijkstra shortest path algorithm with effective diagonal pruning as well as structure inference algorithm, which conforms a complex solution to symbolic music structure analysis. Since the proposed algorithms operate on final self-similarity matrices, it would be interesting to see how our algorithm would perform on audio-based self-similarity images.

The proposed method works well for both classical and popular music, and our quantitative comparison against a handful of audio-based algorithms suggests, that it is very beneficial to use symbolic representations for this task. They are cleaner, they provide more complex and high-level information that allows for more musicologically meaningful analyses. However, our evaluation did not indicate if the advantage of our method comes from the use of symbolic data or the method itself. We know that the use of symbolic information helps, but we do not know to which extent. Porting our method to audio domain and comparison of on the same testbed would be a possible solution to this issue, which would provide far more fair comparison between structure analysis systems. It is also possible that some audio-based methods that can be ported to symbolic domain would yield a better score, comparing to our system.

Chapter 7

Conclusions

7.1 Presented Work

In this dissertation we have presented a number of applications employing n-gram-based techniques to symbolically represented music. Symbolic representations, which include digitized music scores as well as MIDI protocol, are capable of delivering clear and precise information about the actual music events, unlike audio which often combines in a single waveform a number of different sound sources playing several notes at the same time. As it was shown in the preliminary studies, music in its symbolic form demonstrates resemblance to natural languages and this parallel can be used to aid music research with techniques founded upon computational linguistics and information retrieval. The popularity of audio-based methods results from the fact that this representation is much more common. However, there is strong demand for enhancing audio data with symbolic annotations, and if this happens, the proposed techniques could be implemented in a wider context.

Statistical features of symbolic music corpora. Since n-gram-based methods are often used in various computer science domains that typically deal with text, like Information Retrieval, Natural Language Processing or Text Data Mining, we have introduced and performed extensive analysis of statistical features of various symbolic music corpora and compared them to a few text corpora, identifying a number of similarities between natural languages and symbolic music. This justified porting directly n-gram based techniques employing bag-of-terms assumption, used extensively in text processing domain, to certain tasks that both domains have in common, like document classification or retrieval.

Composer classification. For the document classification analysis, we have analyzed three labelled corpora, five similarity measures, several feature types, influence of forced balanced training and extensive range of n-gram lengths. We found that most of the approaches we analyzed, when properly parametrized, can give very good results, on par with other state-of-the art data mining techniques and greatly outperforming humans in composer recognition.

Symbolic music similarity. To evaluate symbolic music similarity measures that have text origin, we have participated in 2011 MIREX symbolic melodic similarity shared task. Our submission consisted of several ranking algorithms featuring four different similarity measures. The results confirmed our previous findings from classification task, that one can obtain similarly good results with any kind of similarity measure if it is properly defined in terms of parameters, and that one of the most important parameter is n-gram length used while creating n-gram features. It was also encouraging that our simple and scalable solutions achieved relatively good overall scores, falling closely behind top performing tailored approaches.

Evolutionary approach to melodies generation. N-gram approach to symbolic music is not just limited to tasks analogue to the tasks in text processing domain. Our next challenge was to implement a genetic algorithm (GA), that would generate pleasant melodies. The goal was to create a fully automatic system, so we have devised a way of assessing pleasantness of generated melodies based on a corpus of existing, pleasant music. The assumption is that if a melody in question is pleasant, it contains subsequences of notes that are important components in our reference corpus. This approach allowed us to test extensively how various evolutionary computation frameworks react to symbolic music specificity and our assumptions. We have found, that traditional generational GA is not capable of controlling the length of the generated melodies, which we called the ‘note bloat’ phenomenon, resembling ‘code bloat’ known from genetic programming. Other evolutionary frameworks, like steady-state GA or multi-objective Pareto optimization were capable of delivering

better quality results. Although we did not perform any quantitative analysis of pleasantness of the results (this would be achievable only through a user study), we have found that obtained melodies kept melodic and harmonic logic, which was not directly imposed by our corpus-based fitness function.

Symbolic music visualization. This was followed by our approach to music content-based visualization and related to it — automatic music structure analysis, which is a timely and interesting topic of current research in the area of Computational Musicology (or Music Information Retrieval). While most of current efforts focus on audio data, our solution to these problems utilizes symbolic data and, again, n-gram based approach.

The proposed visualization system takes MIDI files as input, which are required to have logical channels, representing different voices or instruments separated. The algorithm computes one visualization layer for each pair of channels, looking for similarities between different fragments. The result is displayed on a two-dimensional image, which is typically referred to as self-similarity matrix, visualizing the entire piece with its structural, melodic and textural features visible in a single glimpse, without the need of referring to the actual piece for this information. The system is semi-automatic, as it allows to manually override certain aspects to isolate specific motives or filter layers to reduce clutter. The system has been implemented as an installable program allowing to track both playback and visualization simultaneously to demonstrate its capabilities.

Structural analysis of symbolic music. One of the positive outcomes from the visualization system was the fact that, despite the lack of direct design decisions to focus on visualizing music structure, it was clear from the resulting images, what was the structure of the piece in question. Consequently, the proposed approach to automatic music structure analysis was based on our previous visualization technique, and the principle of creating a self-similarity matrix. This technique is used widely in audio domain. The main contribution of this project was to introduce a method

of measuring similarity between more than one consecutive tracks and to utilize local context of symbolic events. These improvements show that the analysis based on symbolic representation can be more thorough comparing to audio-based analysis, where those low-level music events, like note events, while easily perceived by humans, are typically blended together, prohibiting its accurate detection.

The application of our visualization method to aid automatic music structure analysis required flattening all layers to a single layer where each point of the similarity matrix represented similarity of all tracks between two excerpts within a given piece. It also required further analysis of the similarity measures used in creation of those visualizations to better reveal the underlying structure. The analyzed similarity measures included various text-based techniques, both order-aware and order-free, and we have chosen cosine similarity as best reflecting the desired features. Additional steps consisted of diagonal pruning and calculation of the novelty helper function, which is a higher-order version of the commonly used novelty function in audio domain. The main algorithm implements dynamic programming to find the shortest path on the resulting self-similarity matrix, from the beginning to the end of the analyzed piece. The modification to the original shortest path algorithm allowing the walker to make certain jumps enabled detection of section borders so that the shortest path through the matrix indicated its structure. The comparison with a number of audio based algorithms on a corpus of matching audio and MIDI files along with experts border annotations showed a significant advantage of our method in both precision and recall of correctly assigned borders. However, the answer whether this gap is caused by a superior structure detection algorithm or just simply by employing rich, symbolic music information remained unanswered.

7.2 Impact and Implications

Stephen Downie in his retrospection of MIREX events [42] writes about the glass ceiling effect, where we see very little advancement of current state-of-the art solutions to known problems. The performance of those solutions are far from perfect, while

humans perform such tasks routinely. It would require radically different approaches to current problems in order to shatter this glass ceiling.

Here comes the possible role of symbolic data. Unlike in audio, symbolic music building blocks — notes, and other sound events, are on par what music actually consists of, as far as human perception is concerned. Perhaps audio is a too low-level representation containing basic components (noise bands, tones of certain frequency), which rarely seem to reflect what its needed to solve a certain musical problem. Obviously, adding other levels of aligned data to each recorded piece requires some extra work, but with the help of accurate MIDI-audio alignment techniques (also known as computer accompaniment [31]), it can be done relatively easily. The need of this multi-level analysis was suggested by Vincent et al. [204], where not only audio and symbolic information would be contained within such “intelligent audio” piece, but other layers, e.g., harmonics, structure, performance markings, would be included as well.

Typically, the use of symbolic representations in computational musicology is limited to indexing and to search related problems. This does not have to be the case. We have demonstrated that symbolic data can be a useful material for generation, visualization and various analysis tasks, providing with extra opportunities, like context-aware analysis. Moreover, tasks like music visualization or structure analysis approached in this thesis did not originate from text-related background. This could potentially inspire a move in the other direction — solutions to text-related problems using music-inspired techniques.

7.3 Future Work

Several extensions of the research presented in this thesis can be drawn directly from our results. Our evolutionary approach to melodies generation would benefit from human evaluation of the final results, done through a user study. The system itself could be expanded to generate bigger fragments or entire pieces from smaller building blocks, such as generated melodies. Proposed visualization scheme would also benefit

from a user study, where more and less important aspects of our visualization approach would be identified. The system could be expanded to form a visual analytics tool for musicological analysis or to create a computer-aided composing system for professionals. Our structure analysis algorithm would benefit from further quantitative analysis, where it would be ported to operate on audio data with the same modified Dijkstra path-finding algorithm, or with other algorithms being ported to work with symbolic data, to identify what is the sole benefit of using symbolic data over audio for this problem, and what results from the actual difference in algorithm design.

We can also draw other conclusions for future work with more general consequences, which all revolves around defining new standards incorporating new layers of information with audio data. But to succeed, one has to convince big companies that deal with content creation, like Sony Music, or media content distributors, like Apple iTunes, that everybody, including them, would benefit from the new applications that this new unified music format would make possible, let alone music search and cataloguing possibilities for large music repositories. Other functionalities that this new format would instantly provide includes, for example, automatic karaoke music generation or intelligent music navigation (similar to Goto's SmartMusicKIOSK [64]). The new format and the new data would allow for even higher-level of analysis. We have seen many parallels between natural languages and symbolic music. One of them lies between language's grammars and music harmony rules. High-level syntactic and semantic analysis is at the core of domains like Natural Language Processing and there is no reason, why it could not be the case with computational music, especially that we already have such theoretical foundations in the form of Shenkerian analysis [10, 111].

Bibliography

- [1] Greg Aloupis, Thomas Fevens, Stefan Langerman, Tomomi Matsui, Antonio Mesa, Yurai Nuñez, David Rappaport, and Godfried Toussaint. Algorithms for computing geometric measures of melodic similarity. *Comput. Music J.*, 30(3):67–76, September 2006. Cited at [51]
- [2] Tue Haste Andersen. Mixxx: towards novel DJ interfaces. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*, NIME '03, pages 30–35, Singapore, Singapore, 2003. National University of Singapore. Cited at [179]
- [3] Kjell Bäckman and Palle Dahlstedt. A generative representation for the evolution of jazz solos. In Mario Giacobini, Anthony Brabazon, Stefano Cagnoni, Gianni Di Caro, Rolf Drechsler, Anik Ekrt, Anna Esparcia-Alczar, Muddassar Farooq, Andreas Fink, Jon McCormack, Michael O'Neill, Juan Romero, Franz Rothlauf, Giovanni Squillero, A. Uyar, and Shengxiang Yang, editors, *Applications of Evolutionary Computing*, volume 4974 of *Lecture Notes in Computer Science*, pages 371–380. Springer Berlin / Heidelberg, 2008. Cited at [138]
- [4] David Bainbridge and Tim Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35:95–121, 2001. 10.1023/A:1002485918032. Cited at [8]
- [5] Luke Barrington, Antoni B. Chan, and Gert Lanckriet. Dynamic texture models of music. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 1589–1592, April 2009. Cited at [202]
- [6] Jerome Barthelemy and Alain Bonardi. Similarity in computational music: a musicologist's approach. In *Web Delivering of Music, 2001. Proceedings. First International Conference on*, pages 107–113, November 2001. Cited at [41]
- [7] Pierfrancesco Bellini, Jrme Barthelemy, Paolo Nesi, and Giorgio Zoia. A proposal for the integration of symbolic music notation into multimedia frameworks. In *Web Delivering of Music, 2004. WEDELMUSIC 2004. Proceedings of the Fourth International Conference on*, pages 36–43, September 2004. Cited at [3]
- [8] Pierfrancesco Bellini, Paolo Nesi, and Giorgio Zoia. Symbolic music representation in mpeg. *IEEE MultiMedia*, 12:42–49, October 2005. Cited at [3, 156]
- [9] Tony Bergstrom, Karrie Karahalios, and John C. Hart. Isochords: visualizing structure in music. In *GI '07: Proceedings of Graphics Interface 2007*, pages 297–304, New York, NY, USA, 2007. ACM. Cited at [157, 159]

- [10] David Carson Berry. *A topical guide to Schenkerian literature : an annotated bibliography with indices*. Pendragon Press, 2004. Cited at [217]
- [11] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of ISMIR 2011 (International Conference on Music Information Retrieval)*, ISMIR11, 2011. Cited at [67]
- [12] John A. Biles. GenJam: A genetic algorithm for generating jazz solos. In *Proceedings of the 1994 International Computer Music Conference*, pages 131–137, Aarhus, Denmark, 1994. Cited at [56, 137]
- [13] William P. Birmingham, Roger B. Dannenberg, Gregory H. Wakefield, Mark A. Bartsch, David Bykowski, Dominic Mazzonia, Colin Meek, Maureen Melody, and William Rand. MUSART: Music Retrieval Via Aural Queries. In *Proceedings of ISMIR 2001 (International Symposium on Music Information Retrieval)*, ISMIR01, 2001. Cited at [181]
- [14] Rens Bod. Probabilistic grammars for music. In *Proceedings of the Belgian-Dutch Conference on Artificial Intelligence*, 2001. Cited at [55]
- [15] Rens Bod. A unified model of structural organization in language and music. *JAIR*, 17(1):289–308, 2002. Cited at [55, 61]
- [16] Guillaume Boutard, Samuel Goldszmidt, and Geoffroy Peeters. Browsing inside a music track, the experimentation case study. In *Workshop on LSAS*, 2006. Cited at [178]
- [17] Giuseppe Buzzanca. A supervised learning approach to musical style recognition. In *Proceedings of International Computer Music Conference*, 2002. Cited at [65]
- [18] Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar F. Zaidan. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation, WMT '11*, pages 22–64, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. Cited at [32]
- [19] Emilios Cambouropoulos. Musical parallelism and melodic segmentation: A computational approach. *Music Perception*, 23(3):249–268, 2006. Cited at [180]
- [20] Emilios Cambouropoulos, Maxime Crochemore, Costas Iliopoulos, Laurent Mouchard, and Yoan Pinzon. Algorithms for computing approximate repetitions in musical sequences. In *Proceedings of the 10th Australasian Workshop On Combinatorial Algorithms*, pages 129–144, 1999. Cited at [47]
- [21] Emilios Cambouropoulos, Maxime Crochemore, Costas Iliopoulos, Laurent Mouchard, and Yoan Pinzon. Algorithms for computing approximate repetitions in musical sequences. *International Journal of Computer Mathematics*, 79(11):1135–1148, 2002. Cited at [47]

- [22] Michael Clausen, Roland Engelbrecht, Dirk Meyer, and Jürgen Schmitz. Proms: A web-based tool for searching in polyphonic music. In *Proceedings of ISMIR (International Symposium on Music Information Retrieval)*, ISMIR00, 2000. Cited at [50, 52]
- [23] Raphaël Clifford and Costas S. Iliopoulos. Approximate string matching for music analysis. *Soft Computing — A Fusion of Foundations, Methodologies and Applications*, 8:597–603, 2004. 10.1007/s00500-004-0384-5. Cited at [47]
- [24] Tom Collins, Jeremy Thurlow, Robin Laney, Alistair Willis, and Paul Garthwaite. A comparative evaluation of algorithms for discovering translational patterns in baroque keyboard works. In *Proceedings of the 11th International Society for Music Information Retrieval Conference 2010*, ISMIR 2010, pages 3–8, 2010. Cited at [64]
- [25] Matthew Cooper and Jonathan Foote. Summarizing popular music via structural similarity analysis. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 127–130, October 2003. Cited at [178]
- [26] David Cope. Computer modeling of musical intelligence in EMI. *Comp. Music Journ.*, 16(2):69–83, 1992. Cited at [8, 61]
- [27] Maxime Crochemore, Costas S. Iliopoulos, Thierry Lecroq, and Yoan J. Pinzon. Approximate string matching in musical sequences. In *Proceedings of the Prague Stringology Conference*, pages 26–36, 2001. Cited at [47]
- [28] Roger B. Dannenberg and Masataka Goto. Music structure analysis from acoustic signals. In David Havelock, Sonoko Kuwano, and Michael Vorländer, editors, *Handbook of Signal Processing in Acoustics*, pages 305–331. Springer: New York, 2009. Cited at [179, 180]
- [29] Roger B. Dannenberg and Ning Hu. Pattern discovery techniques for music audio. *Journal of New Music Research*, 32(2):153–163, 2003. Cited at [180, 181]
- [30] Roger B. Dannenberg and Ning Hu. Understanding search performance in query-by-humming systems. In *Proceedings of ISMIR 2004 (International Conference on Music Information Retrieval)*, ISMIR04, 2004. Cited at [49, 53]
- [31] Roger B. Dannenberg and Christopher Raphael. Music score alignment and computer accompaniment. *Commun. ACM*, 49(8):38–43, August 2006. Cited at [7, 32, 216]
- [32] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945. Cited at [189]

- [33] Shyamala Doraisamy. *Polyphonic Music Retrieval: The N-gram Approach*. PhD thesis, University of London, 2004. Cited at [3, 33, 39, 44, 53, 64, 65, 86]
- [34] Shyamala Doraisamy and Stefan R uger. An approach towards a polyphonic music retrieval system. In *Proceedings of ISMIR 2001 (International Conference on Music Information Retrieval)*, ISMIR01, 2001. Cited at [53, 65]
- [35] Shyamala Doraisamy and Stefan R uger. A comparative and fault-tolerance study of the use of n-grams with polyphonic music. In *Proceedings of ISMIR 2002 (International Conference on Music Information Retrieval)*, ISMIR02, 2002. Cited at [53]
- [36] Shyamala Doraisamy and Stefan R uger. Robust polyphonic music retrieval with n-grams. *J. Intell. Inf. Syst.*, 21(1):53–70, 2003. Cited at [53]
- [37] Shyamala Doraisamy and Stefan R uger. A polyphonic music retrieval system using n-grams. In *Proceedings of ISMIR 2004 (International Conference on Music Information Retrieval)*, ISMIR04, 2004. Cited at [53]
- [38] J. Stephen Downie. Informetrics and music information retrieval: An informetric examination of a folksong database. In *Proceedings of the 26th Annual Conference of the Canadian Association for Information Science*, CAIS98, 1998. Cited at [40, 53]
- [39] J. Stephen Downie. Music information retrieval. *Annual Review of Information Science and Technology*, 37:295–340, 2003. Cited at [9, 11]
- [40] Stephen Downie and Michael Nelson. Evaluation of a simple and effective music information retrieval method. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 73–80, New York, NY, USA, 2000. ACM. Cited at [53]
- [41] Stephen Downie, J. *Evaluating a simple approach to music information retrieval: Conceiving melodic n-grams as text*. PhD thesis, University of Western Ontario, 1999. Cited at [3, 53, 63, 64, 65, 68, 73, 86, 94, 95]
- [42] Stephen J. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29:247–255, 2008. Cited at [12, 59, 215]
- [43] Stephen J. Downie. Mirex 2011 evaluation results. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2011. Cited at [7, 29, 156]
- [44] Ben Duane and Bryan Pardo. Streaming from MIDI using Constraint Satisfaction Optimization and Sequence Alignment. In *Proceedings of the International Computer Music Conference 2009, ICMC 2009*, pages 1–9, 2009. Cited at [32, 33, 64]

- [45] Jean-Pierre Eckmann, Sylvie Oliffson-Kamphorst, and David Ruelle. Recurrence plots of dynamical systems. *EPL (Europhysics Letters)*, 4(9):973–977, 1987. Cited at [156]
- [46] Édouard Gilbert and Darrell Conklin. A probabilistic context-free grammar for melodic reduction. In *International Workshop on Artificial Intelligence and Music at Twentieth International Joint Conference on Artificial Intelligence, IJCAI-07*, 2007. Cited at [55]
- [47] Ronald Engelbrecht. Statistical comparison measures for searching in melody databases. Technical report, Communication Systems and Algorithms Group, University of Bonn, 2002. Cited at [55]
- [48] Pascal Ferraro and Pierre Hanna. MIREX Symbolic Music Similarity. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2006. Cited at [119]
- [49] Pascal Ferraro, Pierre Hanna, Julien Allali, and Matthias Robine. MIREX Symbolic Music Similarity. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2007. Cited at [47, 119]
- [50] Michael Fingerhut. Real music libraries in the virtual future: for an integrated view of music and music information. *Digitale Bibliotheken voor muzikale audio*, pages 73–81, 2005. Cited at [9, 10, 11]
- [51] Jonathan Foote. Visualizing music and audio using self-similarity. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 77–80, New York, NY, USA, 1999. ACM. Cited at [157, 158, 161, 180, 183]
- [52] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 1, pages 452–455, 2000. Cited at [180, 197]
- [53] Cristian Francu and Craig G. Nevill-manning. Distance metrics and indexing strategies for a digital library of popular music. In *in proc. IEEE International Conference on Multimedia and Expo (II)*, 2000. Cited at [65]
- [54] Christian Fremerey, Meinard Müller, Frank Kurth, and Michael Clausen. Automatic mapping of scanned sheet music to audio recordings. *Proceedings of the ISMIR, Philadelphia, USA*, pages 413–418, 2008. Cited at [7]
- [55] Bruce Fries and Marty Fries. *Digital audio essentials*. O'Reilly digital studio. O'Reilly, 2005. Cited at [22]
- [56] Joe Futrelle and J. Stephen Downie. Interdisciplinary Research Issues in Music Information Retrieval: ISMIR 2000-2002. *Journal of New Music Research*, 32(2):121–131, 2003. Cited at [3, 4, 5, 6, 7, 10]

- [57] Evgeniy Gabrilovich. Wikipedia preprocessor. <http://sourceforge.net/projects/wikiprep>, November 2012. Cited at [73]
- [58] Andrew Gartland-Jones. MusicBlox: A real-time algorithmic composition system incorporating a distributed interactive genetic algorithm. In G.R. Raidl, editor, *Proceedings: EVOMusArt 2003. Applications of Evolutionary Computing*, volume 2611, pages 490–501. Springer Berlin / Heidelberg, 2003. Cited at [138, 139, 140]
- [59] Zong Geem and Jeong-Yoon Choi. Music composition using harmony search algorithm. In Mario Giacobini, editor, *Applications of Evolutionary Computing*, volume 4448 of *Lecture Notes in Computer Science*, pages 593–600. Springer Berlin / Heidelberg, 2007. Cited at [139]
- [60] Karl Wilson Gehrrens. *Music Notation And Terminology*. The A.S. Barnes company, 1914. Cited at [24]
- [61] Carlos Gómez, Soraya Abad-Mota, and Edna Ruckhaus. An analysis of the Mongeau-Sankoff Algorithm for music information retrieval. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2007. Cited at [50, 119]
- [62] Michael M. Goodwin and Jean Laroche. A dynamic programming approach to audio segmentation and speech/music discrimination. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 4, pages 309–312, May 2004. Cited at [182]
- [63] Masataka Goto. A chorus-section detecting method for musical audio signals. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, volume 5, pages 437–440, April 2003. Cited at [178, 181]
- [64] Masataka Goto. SmartMusicKIOSK: music listening station with chorus-search function. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, pages 31–40, New York, NY, USA, 2003. ACM. Cited at [217]
- [65] Masataka Goto. Active music listening interfaces based on signal processing. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages 1441–1444, April 2007. Cited at [4]
- [66] Masataka Goto and Takayuki Goto. Musicream: New music playback interface for streaming, sticking, sorting, and recalling musical pieces. In *Proceedings of ISMIR 2005 (International Conference on Music Information Retrieval)*, ISMIR05, 2005. Cited at [5]

- [67] Maarten Grachten, Josep Lluís Arcos, and Ramon Lóopez de Máantaras. A comparison of different approaches to melodic similarity. In Christina Anagnostopoulou, M. Ferrand, and Alan Smaill, editors, *Music and Artificial Intelligence: Second International Conference*, ICMAI 2002, Edinburgh, Scotland, UK, September 2002. Cited at [40, 55, 61]
- [68] Maarten Grachten, Josep Lluís Arcos, and Ramon Lóopez de Máantaras. Melodic similarity: Looking for a good abstraction level. In *Proceedings of ISMIR 2004 (International Conference on Music Information Retrieval)*, ISMIR04, 2004. Cited at [42, 49, 55, 61]
- [69] Maarten Grachten, Josep Lluís Arcos, and Ramon Lóopez de Máantaras. Melody retrieval using the implication/realization model. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2005. Cited at [49, 55, 61, 119]
- [70] Masatoshi Hamanaka, Keiji Hirata, and Satoshi Tojo. Implementing a generative theory of tonal music. *Journal of New Music Research*, 35:249–277, 2006. Cited at [8, 54, 60]
- [71] Pierre Hanna, Matthias Robine, and Pascal Ferraro. Visualisation of musical structure by applying improved editing algorithms. In *Proceedings of the International Computer Music Conference (ICMC), Belfast, Northern Ireland*, volume 62, pages 80–81, 2008. Cited at [160, 180]
- [72] Steven Harford. Automatic segmentation, learning and retrieval of melodies using a self-organizing neural network. In *Proceedings of ISMIR 2003 (International Conference on Music Information Retrieval)*, ISMIR03, 2003. Cited at [56]
- [73] Choochart Haruechaiyasak, Sarawoot Kongyoung, and Matthew Dailey. A comparative study on Thai word segmentation approaches. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on*, volume 1, pages 125–128, May 2008. Cited at [123]
- [74] Walter B. Hewlett. A base-40 number-line representation of musical pitch notation. *Musikometrika*, 4:1–14, 1992. Cited at [37]
- [75] Ruben Hillewaere, Bernard Manderick, and Darrel Conklin. String quartet classification with monophonic models. In *Proceedings of the 11th International Society for Music Information Retrieval Conference 2010*, ISMIR 2010, pages 537–542, 2010. Cited at [70, 104, 106, 109, 116]
- [76] Ruben Hillewaere, Bernard Manderick, and Darrell Conklin. String methods for folk tune genre classification. In *Proceedings of ISMIR 2012 (International Conference on Music Information Retrieval)*, ISMIR12, 2012. Cited at [98]

- [77] Keiji Hirata and Shu Matsuda. Interactive music summarization based on generative theory of tonal music. *Journal of New Music Research*, 32:165–177, March 2003. Cited at [54]
- [78] Henkjan Honing. On the growing role of observation, formalization and experimental method in musicology. *Empirical Musicology Review*, 1(1):2–6, January 2006. Cited at [7]
- [79] Holger H. Hoos, Kai Renz, and Marko Göorg. GUIDO/MIR — an experimental musical information retrieval system based on GUIDO music notation. In *Proceedings of ISMIR 2001 (International Symposium on Music Information Retrieval)*, ISMIR01, 2001. Cited at [52, 55]
- [80] Ning Hu and Roger B. Dannenberg. A comparison of melodic database retrieval techniques using sung queries. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, JCDL '02, pages 301–307, New York, NY, USA, 2002. ACM. Cited at [47, 49]
- [81] Ning Hu, Roger B. Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 185–188, October 2003. Cited at [7]
- [82] David Huron. Music information processing using the humdrum toolkit: Concepts, examples, and lessons. *Comput. Music J.*, 26:11–26, July 2002. Cited at [25, 37]
- [83] Avid Technology Inc. Sibelius - the leading music composition and notation software. <http://www.sibelius.com>, March 2011. Cited at [25]
- [84] MakeMusic Inc. Finale music composing and notation software. <http://www.finalemusic.com>, March 2011. Cited at [25]
- [85] Eric Isaacson. What you see is what you get: on visualizing music. In *Proceedings of ISMIR 2005 (International Conference on Music Information Retrieval)*, ISMIR05, 2005. Cited at [6, 12, 24, 157, 159, 196]
- [86] Paul Jaccard. *Distribution de la Flore Alpine: dans le Bassin des dranses et dans quelques régions voisines*. Rouge, 1901. Cited at [189]
- [87] Bruce Jacob. Composing with genetic algorithms. In *Proceedings of the 1995 International Computer Music Conference*, pages 452–455, 1995. Cited at [56, 138]
- [88] Tristan Jehan. Perceptual segment clustering for music description and time-axis redundancy cancellation. In *Proceedings of ISMIR 2004 (International Conference on Music Information Retrieval)*, ISMIR04, 2004. Cited at [179]

- [89] Tristan Jehan and Davis DesRoches. Analyzer documentation. Technical report, The Echo Nest Corporation, September 2011. Cited at [202]
- [90] Kristoffer Jensen. Multiple scale music segmentation using rhythm, timbre, and harmony. *EURASIP Journal on Advances in Signal Processing*, 2007(1):159–159, 2007. Cited at [182, 183]
- [91] Kristoffer Jensen, Jieping Xu, and Martin Zachariassen. Rhythm-based segmentation of popular chinese music. In *Proceedings of ISMIR 2005 (International Conference on Music Information Retrieval)*, ISMIR05, 2005. Cited at [179, 182]
- [92] Brad Johanson and Riccardo Poli. GP-Music: An interactive genetic programming system for music generation with automated fitness raters. Technical Report CSR-98-13, School of Computer Science, The University of Birmingham, Birmingham, UK, 1998. Cited at [56, 138]
- [93] Vlado Keselj, Fuchun Peng, Nick Cercone, and Calvin Thomas. N-gram-based author profiles for authorship attribution. In *Proc. of the PACLING03 Conf.*, pages 255–264, 2003. Cited at [61, 105, 123, 125, 189]
- [94] Yaser Khalifa and Robert Foster. A two-stage autonomous evolutionary music composer. In Franz Rothlauf, Jrgen Branke, Stefano Cagnoni, Ernesto Costa, Carlos Cotta, Rolf Drechsler, Evelyne Lutton, Penousal Machado, Jason Moore, Juan Romero, George Smith, Giovanni Squillero, and Hideyuki Takagi, editors, *Applications of Evolutionary Computing*, volume 3907 of *Lecture Notes in Computer Science*, pages 717–721. Springer: Berlin / Heidelberg, 2006. Cited at [139, 140]
- [95] Jürgen Kilian and Holger H. Hoos. MusicBLAST — gapped sequence alignment for MIR. In *Proceedings of ISMIR 2004 (International Conference on Music Information Retrieval)*, ISMIR04, 2004. Cited at [49]
- [96] Julian Kupiec. A trellis-based algorithm for estimating the parameters of a hidden stochastic context-free grammar. In *HLT '91: Proc. of the workshop on Speech and Natural Language*, pages 241–246, Morristown, NJ, USA, 1991. Assoc. for Computational Linguistics. Cited at [61]
- [97] Frank Kurth, Meinard Müller, Christian Fremerey, Yoon-ha Chang, and Michael Clausen. Automated synchronization of scanned sheet music with audio recordings. *Proc. ISMIR, Vienna, Austria*, pages 261–266, 2007. Cited at [7]
- [98] International Music Information Retrieval Systems Evaluation Laboratory. Mirex 2007 challenge on symbolic melodic similarity. http://www.music-ir.org/mirex/wiki/2007:Symbolic_Melodic_Similarity, 2007. Cited at [67]

- [99] International Music Information Retrieval Systems Evaluation Laboratory. Mirex challenge on symbolic melodic similarity. http://www.music-ir.org/mirex/wiki/Symbolic_Melodic_Similarity, 2010. Cited at [13]
- [100] International Music Information Retrieval Systems Evaluation Laboratory. Mirex 2011 challenge on symbolic melodic similarity. http://www.music-ir.org/mirex/wiki/2011:Symbolic_Melodic_Similarity_Results, August 2011. Cited at [131]
- [101] Mika Laitinen and Kjell Lemström. Geometric algorithms for melodic similarity. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2010. Cited at [51, 118]
- [102] Paul Lansky. Reflections on spent time. In *Proceedings of the International Computer Music Conference 2009, ICMC 2009*, pages 561–568, 2009. Cited at [1]
- [103] Edwin Hui Hean Law and Somnuk Phon-Amnuaisuk. Towards music fitness evaluation with the hierarchical SOM. In *Proceedings of the 2008 conference on Applications of evolutionary computing, Evo’08*, pages 443–452. Springer-Verlag: Berlin, Heidelberg, 2008. Cited at [56, 139]
- [104] John Lazzaro and John Wawrzynek. RTP Payload Format for MIDI. Technical report, Internet Engineering Task Force, June 2011. RFC 6295. Cited at [26]
- [105] Jin Ha Lee and J. Stephen Downie. Survey of music information needs, uses, and seeking behaviours: Preliminary findings. In *Proceedings of ISMIR 2004 (International Conference on Music Information Retrieval)*, 2004. Cited at [4, 5, 6, 8]
- [106] Juwan Lee, Seokhwan Jo, and Chang D. Yoo. Coded melodic contour model. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2011. Cited at [131, 132]
- [107] Kjell Lemström. *String matching Techniques for Music Retrieval*. PhD thesis, University of Helsinki, Helsinki, Finland, 2000. Cited at [3, 64]
- [108] Kjell Lemström, Niko Mikkilä, Veli Mäkinen, and Esko Ukkonen. String matching and geometric algorithm for melodic similarity. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2005. Cited at [49, 50, 118, 119]
- [109] Kjell Lemström, Niko Mikkilä, Veli Mäkinen, and Esko Ukkonen. Sweepline and recursive geometric algorithms for melodic similarity. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2006. Cited at [50, 118]
- [110] Kjell Lemström and Anna Pienimäki. On comparing edit distance and geometric frameworks in content-based retrieval of symbolically encoded polyphonic music. *Musicae Scientiae*, 11(1 suppl):135–152, 2007. Cited at [51]

- [111] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. The MIT Press, 1983. Cited at [8, 54, 60, 217]
- [112] Mark Levy and Mark Sandler. Structural segmentation of musical audio by constrained clustering. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2):318–326, February 2008. Cited at [202]
- [113] Ming Li, Xin Chen, Xin Li, Bin Ma, and P.M.B. Vitanyi. The similarity metric. *Information Theory, IEEE Transactions on*, 50(12):3250–3264, December 2004. Cited at [101]
- [114] Ming Li and Ronan M. Sleep. Melody classification using a similarity metric based on Kolmogorov complexity. In *Proceeding of Conference on Sound and Music Computing*, 2004. Cited at [52, 53, 98, 100, 101]
- [115] Anna Lubiw and Luke Tanur. Pattern matching in polyphonic music as a weighted geometric translation problem. In *Proceedings of ISMIR 2004 (International Conference on Music Information Retrieval)*, ISMIR04, 2004. Cited at [51]
- [116] Penousal Machado, Juan Romero, Mara Santos, Amílcar Cardoso, and Bill Manaris. Adaptive critics for evolutionary artists. In Gunther Raidl, Stefano Cagnoni, Jrgen Branke, David Corne, Rolf Drechsler, Yaochu Jin, Colin Johnson, Penousal Machado, Elena Marchiori, Franz Rothlauf, George Smith, and Giovanni Squillero, editors, *Applications of Evolutionary Computing*, volume 3005 of *Lecture Notes in Computer Science*, pages 437–446. Springer: Berlin / Heidelberg, 2004. Cited at [139]
- [117] Namunu C. Maddage, Changsheng Xu, Mohan S. Kankanhalli, and Xi Shao. Content-based music structure analysis with applications to music semantics understanding. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, MULTIMEDIA '04, pages 112–119, New York, NY, USA, 2004. ACM. Cited at [178]
- [118] Donncha Ó Maidín. A geometrical algorithm for melodic difference. *Computing in Musicology*, 11:65–72, 1998. Cited at [51]
- [119] Veli Mäkinen, Gonzalo Navarro, and Esko Ukkonen. Algorithms for transposition invariant string matching. Technical Report DCC-2002-5, Department of Computer Science, University of Chile, Santiago, Chile, July 2002. Cited at [46]
- [120] Veli Mäkinen, Gonzalo Navarro, and Esko Ukkonen. Algorithms for transposition invariant string matching (extended abstract). In Helmut Alt and Michel Habib, editors, *STACS 2003*, volume 2607 of *Lecture Notes in Computer Science*, pages 191–202. Springer: Berlin / Heidelberg, 2003. Cited at [46, 49]
- [121] Stephen Malinowski. Music animation machine. <http://www.musanim.com>, October 2010. Cited at [50, 157, 159]

- [122] Bill Manaris, Penousal Machado, Clayton McCauley, Juan Romero, and Dwight Krehbiel. Developing fitness functions for pleasant music: Zipfs law and interactive evolution systems. In Franz Rothlauf, Jürgen Branke, Stefano Cagnoni, David Wolfe Corne, Rolf Drechsler, Yaochu Jin, Penousal Machado, Elena Marchiori, Juan Romero, George D. Smith, and Giovanni Squillero, editors, *Proceedings: EVOMusArt 2005. Applications of Evolutionary Computing*, volume 3449, pages 498–507. Springer: Berlin / Heidelberg, 2005. Cited at [139, 144]
- [123] Bill Manaris, Dallas Vaughan, Christopher Wagner, Juan Romero, and Robert B. Davis. Evolutionary music and the Zipf-Mandelbrot law: developing fitness functions for pleasant music. In *Proceedings of the 2003 international conference on Applications of evolutionary computing, EvoWorkshops'03*, pages 522–534. Springer-Verlag: Berlin, Heidelberg, 2003. Cited at [139, 144]
- [124] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, June 1999. Cited at [61]
- [125] Matija Marolt. A mid-level melody-based representation for calculating audio similarity. In *Proceedings of ISMIR 2006 (International Conference on Music Information Retrieval)*, ISMIR06, 2006. Cited at [7, 12, 14, 179, 180, 181]
- [126] Jon McCormack. Open problems in evolutionary music and art. In et al. Rothlauf, F., editor, *Proceedings: EVOMusArt 2005. Applications of Evolutionary Computing*, volume 3449, pages 428–436. Springer: Berlin / Heidelberg, 2005. Cited at [136]
- [127] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. The million song dataset challenge. In *Proceedings of the 21st International Conference Companion on World Wide Web, WWW '12 Companion*, pages 909–916, New York, NY, USA, 2012. ACM. Cited at [9]
- [128] Colin Meek and William Birmingham. Johnny can't sing: A comprehensive error model for sung music queries. In *Proceedings of ISMIR 2002 (International Conference on Music Information Retrieval)*, ISMIR02, 2002. Cited at [37, 39, 44, 45]
- [129] Massimo Melucci and Nicola Orio. Musical information retrieval using melodic surface. In *Proceedings of the Fourth ACM Conference on Digital Libraries, DL '99*, pages 152–160, New York, NY, USA, 1999. ACM. Cited at [52]
- [130] Massimo Melucci and Nicola Orio. Smile: a system for content-based musical information retrieval environments. In *RIAO 2000 Conference Proceedings, Vol 2*, page 1261, 2000. Cited at [52]
- [131] Xavier Mestres. A BIC-based approach to singer identification. Master's thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2007. Cited at [202]

- [132] Daniel Muellensiefen and Klaus Frieler. Cognitive adequacy in the measurement of melodic similarity: Algorithmic vs. human judgments. *Computing in Musicology*, 13:147–176, 2003. Cited at [40, 42, 107]
- [133] Meinard Müller and Frank Kurth. Towards structural analysis of audio recordings in the presence of musical variations. *EURASIP J. Appl. Signal Process.*, 2007(1):163–163, January 2007. Cited at [182]
- [134] Declan Murphy. Pattern play. In *Additional Proceedings of the 2nd International Conference on Music and Artificial Intelligence*, 2002. Cited at [179, 180]
- [135] Eugene Narmour. *The analysis and cognition of basic melodic structures: The implication-realization model*. University of Chicago Press, 1990. Cited at [8, 50, 54, 61]
- [136] Giovanna Neve and Nicola Orio. Indexing and retrieval of music documents through pattern analysis and data fusion techniques. In *Proceedings of ISMIR 2004 (International Conference on Music Information Retrieval)*, ISMIR04, 2004. Cited at [33, 39, 43, 52]
- [137] NIST. Text retrieval conference. <http://trec.nist.gov/>, January 2012. Cited at [118]
- [138] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, April 1928. Cited at [21]
- [139] Tomasz Oliwa and Markus Wagner. Composing music with neural networks and probabilistic finite-state machines. In *Proceedings of the 2008 EvoWorkshops 2008 on EvoCoMnet, EvoFIN, EvoIASP, EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog*, pages 503–508. Springer-Verlag: Berlin, Heidelberg, 2008. Cited at [56, 139]
- [140] Nicola Orio. Combining multilevel and multi-feature representation to compute melodic similarity. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2005. Cited at [39, 119]
- [141] Nicola Orio. Music retrieval: a tutorial and review. *Found. Trends Inf. Retr.*, 1(1):1–96, January 2006. Cited at [4, 5, 6, 46, 56]
- [142] Nicola Orio and Giovanna Neve. Experiments on segmentation techniques for music documents indexing. In *Proceedings of ISMIR 2005 (International Conference on Music Information Retrieval)*, ISMIR05, 2005. Cited at [4, 53]
- [143] George Papadopoulos and Geraint Wiggins. A genetic algorithm for the generation of jazz melodies. In *Proceedings of the Finnish Conference on Artificial Intelligence (SteP 1998)*, Jyvaskyla, Finland, September 1998. Cited at [138]

- [144] Bryan Pardo and William Birmingham. Encoding timing information for musical query matching. In *Proceedings of ISMIR 2002 (International Conference on Music Information Retrieval)*, ISMIR02, 2002. Cited at [39, 44, 45]
- [145] Bryan Pardo, Jonah Shifrin, and William Birmingham. Name that tune: A pilot study in finding a melody from a sung query. *Journal of the American Society for Information Science and Technology*, 55(4):283–300, 2004. Cited at [47, 52, 55, 65]
- [146] Jouni Paulus and Anssi Klapuri. Music structure analysis using a probabilistic fitness measure and a greedy search algorithm. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(6):1159–1170, August 2009. Cited at [181, 182]
- [147] Jouni Paulus, Meinard Müller, and Anssi Klapuri. Audio-based music structure analysis. In *Proceedings of the 11th International Society for Music Information Retrieval Conference 2010*, ISMIR 2010, pages 625–636, 2010. Cited at [14, 179, 180, 183]
- [148] Geoffroy Peeters. Toward automatic music audio summary generation from signal analysis. In *Proceedings of ISMIR 2002 (International Conference on Music Information Retrieval)*, ISMIR02, 2002. Cited at [182, 183]
- [149] Geoffroy Peeters. Sequence representation of music structure using higher-order similarity matrix and maximum-likelihood approach. In *Proceedings of ISMIR 2007 (International Conference on Music Information Retrieval)*, ISMIR07, 2007. Cited at [181, 197]
- [150] Ewald Peiszer. Automatic audio segmentation: Segment boundary and structure detection in popular music. Master’s thesis, Technischen Universität Wien, Vienna, Austria, 2007. Cited at [202]
- [151] Somnuk Phon-Amnuaisuk, Edwin Hui Law, and Ho Chin Kuan. Evolving music generation with SOM-Fitness genetic programming. In *Proceedings of the 2007 EvoWorkshops 2007 on EvoCoMnet, EvoFIN, EvoIASP, EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog*, pages 557–566. Springer-Verlag: Berlin, Heidelberg, 2007. Cited at [56, 139]
- [152] Jeremy Pickens. A comparison of language modeling and probabilistic text information retrieval approaches to monophonic music retrieval. In *Proceedings of ISMIR (International Symposium on Music Information Retrieval)*, ISMIR00, 2000. Cited at [53]
- [153] Anna Pienimäki. Indexing music databases using automatic extraction of frequent phrases. In *Proceedings of ISMIR 2002 (International Conference on Music Information Retrieval)*, ISMIR02, 2002. Cited at [52]

- [154] Alberto Pinto. Mirex 2007 - graph spectral method. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2007. Cited at [55, 119]
- [155] Alberto Pinto, Reinier H. van Leuken, M. Fatih Demirci, Frans Wiering, and Remco C. Veltkamp. Indexing music collections through graph spectra. In *Proceedings of ISMIR 2007 (International Conference on Music Information Retrieval)*, ISMIR07, 2007. Cited at [55]
- [156] Tim Pohle, Elias Pampalk, and Gerhard Widmer. Evaluation of frequently used audio features for classification of music into perceptual categories. In *Proceedings of the Fourth International Workshop on Content-Based Multimedia Indexing (CBMI'05)*, 2005. Cited at [23]
- [157] Emanuele Pollastri and Giuliano Simoncelli. Classification of melodies by composer with hidden markov models. In *WEDELMUSIC '01: Proceedings of the First International Conference on WEB Delivering of Music (WEDELMUSIC'01)*, pages 88–95, Washington, DC, USA, 2001. IEEE Computer Society. Cited at [65]
- [158] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 275–281, New York, NY, USA, 1998. ACM. Cited at [53]
- [159] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre Marcal, Carlos Guedes, and Jaime Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, pages 1–18, 2012. 10.1007/s13735-012-0004-6. Cited at [8]
- [160] Christoph Riedweg. *Pythagoras; His Life, Teaching, and Influence*. Cornell University Press, 2002. Cited at [2]
- [161] David Rizo and José M. Iñesta. Trees and combined methods for monophonic music similarity evaluation. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2010. Cited at [55, 119]
- [162] David Rizo and José M. Iñesta. Tree-structured representation of melodies for comparison and retrieval. In *Proc. of the 2nd Int. Conf. on Pattern Recognition in Information Systems*, PRIS 2002, pages 140–155, 2002. Cited at [55]
- [163] David Rizo Valero. *Symbolic music comparison with tree data structures*. PhD thesis, Univesidad de Alicante, Alicante, Spain, 2010. Cited at [35, 36, 37, 43, 46, 55]
- [164] Matti P. Rynänen and Anssi P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Comput. Music J.*, 32:72–86, September 2008. Cited at [7, 29]

- [165] Eleanor Selfridge-Field. Conceptual and representational issues in melodic comparison. *Computing in Musicology*, 11:3–64, 1998. Cited at [37, 43]
- [166] Wayne M. Senner. *The Origins of Writing*. University of Nebraska Press, 1991. Cited at [24]
- [167] Johnny F. Serrano and Jose M. Inesta. Music information retrieval through melodic similarity using Hanson intervallic analysis. *Research in Computing Science*, 20:131–142, 2006. Cited at [52]
- [168] Johnny F. Serrano and Jose M. Inesta. Music motive extraction through Hanson intervallic analysis. In *Computing, 2006. CIC '06. 15th International Conference on*, pages 154–160, November 2006. Cited at [52]
- [169] Yu Shiua, Hong Jeongb, and C-C Jay Kuo. Similar segment detection for music structure analysis via Viterbi algorithm. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 789–792, July 2006. Cited at [183]
- [170] Jordan B. L. Smith. A comparison and evaluation of approaches to the automatic formal analysis of musical audio. Master’s thesis, McGill University, 2010. Cited at [201, 202, 203]
- [171] Jordan B. L. Smith, J. Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J. Stephen Downie. Design and creation of a large-scale database of structural annotations. In *Proceedings of ISMIR 2011 (International Conference on Music Information Retrieval)*, ISMIR11, 2011. Cited at [202]
- [172] Jon Snyder and Marti Hearst. ImproViz: visual explorations of jazz improvisations. In *CHI '05: Computer Human Interaction 2005 extended abstracts on Human factors in computing systems*, pages 1805–1808, New York, NY, USA, 2005. ACM. Cited at [157, 159]
- [173] Gilbert A. Soulodre, Theodore Grusec, Michel Lavoie, and Louis Thibault. Subjective evaluation of state-of-the-art two-channel audio codecs. *J. Audio Eng. Soc.*, 46(3):164–177, 1998. Cited at [23]
- [174] Richard Stenzel and Thomas Kamps. Improving content-based similarity measures by training a collaborative model. In *Proceedings of ISMIR 2005 (International Conference on Music Information Retrieval)*, ISMIR05, 2005. Cited at [5]
- [175] Iman S. H. Suyoto and Alexandra L. Uitdenbogerd. Simple efficient n-gram indexing for effective melody retrieval. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2005. Cited at [105, 119]
- [176] Iman S. H. Suyoto and Alexandra L. Uitdenbogerd. Simple orthogonal pitch matching with ioi symbolic music matching. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2010. Cited at [119]

- [177] Luke Tanur. *A Geometric Approach to Pattern Matching in Polyphonic Music*. PhD thesis, Waterloo University, Waterloo, ON, Canada, 2005. Cited at [51]
- [178] David Temperley. *Music and Probability*. The MIT Press, 2007. Cited at [8, 61]
- [179] Belinda Thom. Unsupervised learning and interactive jazz/blues improvisation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 652–657. AAAI Press / The MIT Press, 2000. Cited at [65]
- [180] Nao Tokui and Hitoshi Iba. Music composition with interactive evolutionary computation. *Communication*, 17(2):215–226, 2000. Cited at [56, 138]
- [181] Kazuto Tominaga and Masafumi Setomoto. An artificial-chemistry approach to generating polyphonic musical phrases. In Mario Giacobini, Anthony Brabazon, Stefano Cagnoni, Gianni Di Caro, Rolf Drechsler, Anik Ekrt, Anna Esparcia-Alczar, Muddassar Farooq, Andreas Fink, Jon McCormack, Michael O'Neill, Juan Romero, Franz Rothlauf, Giovanni Squillero, A. Uyar, and Shengxiang Yang, editors, *Applications of Evolutionary Computing*, volume 4974 of *Lecture Notes in Computer Science*, pages 463–472. Springer: Berlin / Heidelberg, 2008. Cited at [139]
- [182] Leo Treitler. *With Voice and Pen*. University of Nebraska Press, 2000. Cited at [24]
- [183] Wei-Ho Tsai, Hung-Ming Yu, and Hsin-Min Wang. A query-by-example technique for retrieving cover versions of popular songs with similar melodies. In *Proceedings of ISMIR 2005 (International Conference on Music Information Retrieval)*, ISMIR05, 2005. Cited at [47]
- [184] Alan M. Turing. Computing, machinery and intelligence. *Mind*, 49:433–460, 1950. Cited at [58]
- [185] Rainer Typke. *Music Retrieval based on Melodic Similarity*. PhD thesis, Utrecht University, Utrecht, The Netherlands, 2007. Cited at [51]
- [186] Rainer Typke, Marc den Hoed, Justin de Nooijer, Frans Wiering, and Remco C. Veltkamp. A ground truth for half a million musical incipits. *Journal of Digital Information Management*, 3:34–39, 2005. Cited at [72, 119]
- [187] Rainer Typke, Remco C. Veltkamp, and Frans Wiering. A measure for evaluating retrieval techniques based on partially ordered ground truth lists. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 1793–1796, July 2006. Cited at [121]

- [188] Rainer Typke, Frans Wiering, and Remco C. Veltkamp. Evaluating the earth mover's distance for measuring symbolic melodic similarity. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2005. Cited at [51, 118]
- [189] Rainer Typke, Frans Wiering, and Remco C. Veltkamp. A survey of music information retrieval systems. In *Proceedings of ISMIR 2005 (International Conference on Music Information Retrieval)*, ISMIR05, 2005. Cited at [4]
- [190] Rainer Typke, Frans Wiering, and Remco C. Veltkamp. MIREX symbolic melody similarity and query by singing/humming. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2006. Cited at [51, 118]
- [191] George Tzanetakis and Perry Cook. Multifeature audio segmentation for browsing and annotation. In *Applications of Signal Processing to Audio and Acoustics, 1999 IEEE Workshop on*, pages 103–106, 1999. Cited at [178]
- [192] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE Transactions on*, 10(5):293–302, July 2002. Cited at [23]
- [193] Alexandra Uitdenbogerd and Justin Zobel. Melodic matching techniques for large music databases. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 57–66, New York, NY, USA, 1999. ACM. Cited at [3, 33, 39, 40, 42, 49, 53, 64, 65, 68, 107, 162]
- [194] Alexandra L. Uitdenbogerd. *Music Information Retrieval Technology*. PhD thesis, RMIT University, Melbourne, Victoria, Australia, 2002. Supervisors — Justing Zobel and Hugh Williams. Cited at [64, 67, 94]
- [195] Alexandra L. Uitdenbogerd. Variations on local alignment for specific query types. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2006. Cited at [119]
- [196] Alexandra L. Uitdenbogerd. N-gram pattern matching and dynamic programming for symbolic melody search. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2007. Cited at [39, 119]
- [197] Alexandra L. Uitdenbogerd and Yaw Wah Yap. Was Parsons right? an experiment in usability of music representations for melody-based music retrieval. In *Proceedings of ISMIR 2003 (International Conference on Music Information Retrieval)*, ISMIR03, 2003. Cited at [40, 43, 45]
- [198] Alexandra L. Uitdenbogerd and Justin Zobel. Music ranking techniques evaluated. *Aust. Comput. Sci. Commun.*, 24(1):275–283, January 2002. Cited at [53]
- [199] Esko Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92(1):191–211, 1992. Cited at [107]

- [200] Esko Ukkonen, Kjell Lemström, and Veli Mäkinen. Geometric algorithms for transposition invariant content-based music retrieval. In *Proceedings of ISMIR 2003 (International Conference on Music Information Retrieval)*, ISMIR03, 2003. Cited at [50]
- [201] Julián Urbano, Juan Lloréns, and Sonia Sánchez-Cuadrado. Local alignment with geometric representations. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2010. Cited at [119]
- [202] Julián Urbano, Juan Lloréns, and Sonia Sánchez-Cuadrado. Sequence alignment with geometric representations. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2011. Cited at [119, 131, 132]
- [203] Julián Urbano, Mónica Marrero, Diego Martín, and Juan Lloréns. Improving the Generation of Ground Truths based on Partially Ordered Lists. In *Proceedings of International Society for Music Information Retrieval Conference*, pages 285–290, 2010. Cited at [120]
- [204] Emmanuel Vincent, Stanisław A Raczynski, Nobutaka Ono, and Shigeki Sagayama. A roadmap towards versatile MIR. In *Proceedings of 2010 International Society for Music Information Retrieval Conference (ISMIR), fMIR Workshop*, pages 662–664, 2010. Cited at [9, 12, 216]
- [205] Avery Wang. An industrial strength audio search algorithm. In *Proceedings of ISMIR 2003 (International Conference on Music Information Retrieval)*, ISMIR03, 2003. Cited at [4]
- [206] Avery Wang. The Shazam music recognition service. *Communications of the ACM*, 49(8):44–48, August 2006. Cited at [4]
- [207] Geraint A. Wiggins, Kjell Lemström, and Davis Meredith. SIA(M)ESE: An algorithm for transposition invariant, polyphonic content-based music retrieval. In *Proceedings of ISMIR 2002 (International Conference on Music Information Retrieval)*, ISMIR02, 2002. Cited at [50]
- [208] Fritz Winckel. *Phänomene des musikalischen Hörens: Ästhetisch-naturwissenschaftliche Betrachtungen*. Dover books T1764. Dover Publications, 1967. Cited at [21, 22]
- [209] Jacek Wołkowicz, Stephen Brooks, and Vlado Kešelj. Midivis: Visualizing music structure via similarity matrices. In *Proceedings of International Computer Music Conference ICMC2009*, pages 53–56, Montreal, Canada, August 2009. Cited at [17, 18, 184, 187, 193]

- [210] Jacek Wołkowicz, Malcolm Heywood, and Vlado Kešelj. Evolving indirectly represented melodies with corpus-based fitness evaluation. In Mario Jacobini, Anthony Brabazon, Stefano Cagnoni, Gianni A. Caro, Anikó Ekárt, Anna Isabel Esparcia-Alcázar, Muddassar Farooq, Andreas Fink, and Penousal Machado, editors, *EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, volume 5484 of *Lecture Notes in Computer Science*, pages 603–608, Tübingen, Germany, April 2009. Springer-Verlag: Berlin, Heidelberg. Cited at [16, 17]
- [211] Jacek Wołkowicz and Vlado Kešelj. Predicting development of research in music based on parallels with natural language processing. In *Proceedings of the 2010 International Society for Music Information Retrieval Conference (ISMIR), fMIR Workshop*, pages 665–667, 2010. Cited at [15, 18]
- [212] Jacek Wołkowicz and Vlado Kešelj. Text information retrieval approach to music information retrieval. In *Music Information Retrieval Evaluation eXchange (MIREX)*, 2011. Cited at [16, 18]
- [213] Jacek Wołkowicz and Vlado Kešelj. Analysis of important factors for measuring similarity of symbolic music using n-gram-based, bag-of-words approach. In Leila Kosseim and Diana Inkpen, editors, *Advances in Artificial Intelligence*, volume 7310 of *Lecture Notes in Computer Science*, pages 230–241. Springer: Berlin / Heidelberg, 2012. Cited at [16, 18, 196]
- [214] Jacek Wołkowicz and Vlado Kešelj. Evaluation of n-gram-based classification approaches on classical music corpora. In *Proceedings to The Fourth International Conference on Mathematics and Computation in Music*, Montreal, Canada, June 2013. Springer-Verlag: Berlin, Heidelberg. Cited at [18]
- [215] Jacek Wołkowicz, Zbigniew Kulka, and Vlado Kešelj. N-gram-based approach to composer recognition. *Archives of Acoustics*, 33(2008)(1):43–55, January 2008. Cited at [18, 61, 65, 67, 108, 190]
- [216] Jacek Michał Wołkowicz. N-gram-based approach to composer recognition. Master’s thesis, Warsaw University of Technology, Warsaw, Poland, 2007. Supervisor — Kulka Zbigniew. Cited at [2, 45, 68, 103, 106, 108, 124, 125]
- [217] Peter Worth and Susan Stepney. Growing music: Musical interpretations of L-Systems. In Franz Rothlauf, Jürgen Branke, Stefano Cagnoni, David Wolfe Corne, Rolf Drechsler, Yaochu Jin, Penousal Machado, Elena Marchiori, Juan Romero, George D. Smith, and Giovanni Squillero, editors, *Proceedings: EVO-MusArt 2005. Applications of Evolutionary Computing*, volume 3449, pages 545–550. Springer: Berlin / Heidelberg, 2005. Cited at [139]
- [218] Keith Wyatt and Carl Schroeder. *Harmony and Theory: A Comprehensive Source for All Musicians*. Essential Concepts. Hal Leonard, 1998. Cited at [39]

- [219] Feng Yahzong, Zhuang Yueting, and Pan Yunhe. Query similar music by correlation degree. In Heung-Yeung Shum, Mark Liao, and Shih-Fu Chang, editors, *Advances in Multimedia Information Processing PCM 2001*, volume 2195 of *Lecture Notes in Computer Science*, pages 885–890. Springer Berlin / Heidelberg, 2001. Cited at [56]
- [220] George Kingsley Zipf. *Human behavior and the principle of least effort: An introduction to human ecology*. Hafner Pub. Co (1972), 1949. Cited at [93]