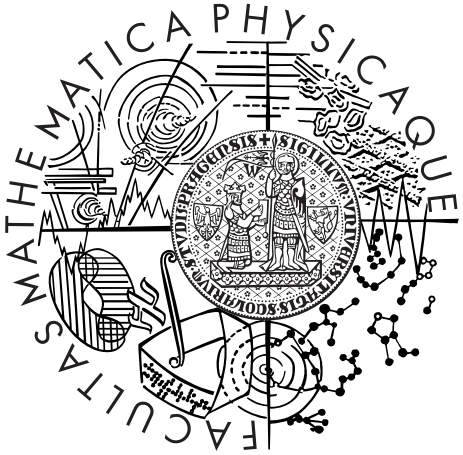
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Jindřich Libovický

Statistical Natural Language Processing Methods in Music Notation Analysis

Institute of Formal and Applied Linguistics

Supervisor of the master thesis: Mgr. Nino Peterek, PhD.

Study program: Computer Science

Specialization: Computational Linguistics

Prague 2013

I would like to thank *Music Files Ltd* – owner of www.mflies.co.uk, *Mr.* Berndt Kreuger – author of Classical Piano Midi Page (www.piano-midi.de) and *Mr. Les Winters* – keeper of The Classical MIDI Collection (www.classicalmidiconnection.com) for allowing me to use their MIDI Collections in this thesis.

I also owe my special thanks to my thesis supervisor, Mgr. Nino Peterek, PhD., for his support, and to all my friends who helped me make the text better readable.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague, 12.4. 2013 .....

Název práce: Statistické metody zpracování přirozených jazyků v analýze notopisu

Autor: Jindřich Libovický

Katedra: Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: Mgr. Nino Peterek, PhD.

Abstrakt: Práce shrnuje dosavadní výzkum v oblasti aplikace statistických metod počítačové lingvistiky při zpracování hudby a vysvětluje teoretické pozadí těchto aplikací. V druhé části práce jsou shrnuty možnosti symbolické extrakce melodie. Byl vytvořen korpus přibližně 400 hodin melodií různých hudebních stylů, který je využit pro trénování statistického modelu melodie založeného na metodách jazykového modelování. V třetí části práce je tento model využit k pokusu vytvořit alternativní metodu extrakce melodie ze zvukového záznamu, která místo běžně používaných heuristik a pravidel využívá model melodie. Systém funguje dobře pouze na jednoduchých vstupních datech, ale na standardních datech ze soutěže MIREX nedosahuje úspěšnosti v současnosti existujících systému. Provedené experimenty s rozpoznáváním melodie pomohly lépe definovat rozdíl mezi tím, jak vypadá průběh frekvence vnímané jako melodie – fyzikální melodie, a jak je melodie vnímána na abstraktní úrovni při symbolickém zápisu – abstraktní melodie.

Klíčová slova: statistické metody NLP, notopis, modelování melodie, rozpoznání melodie

Title: Statistical Natural Language Processing Methods in Musical Notation Analysis

Author: Jindřich Libovický

Department: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Nino Peterek, PhD.

Abstract: The thesis summarizes the research in application of statistical methods of computational linguistics in music processing and explains theoretical background of these applications. In the second part methods of symbolic melody extraction are explored. A corpus of approximately 400 hours of melodies of different music styles was created. A melody model using the language modeling techniques was trained on this corpus. In the third part of the thesis the model is used for an attempt to develop an alternative method of audio melody extraction which uses the melody model instead of commonly used heuristics and rules. The chosen approach works well only on simple input data and produces worse results than the commonly used methods on the MIREX contest data. On the other hand, the experiments help to understand the conceptual between the pitch frequency development – the physical melody – and the melody perceived on an abstract level in the symbolic notation – the symbolic melody.

Keywords: statistical NLP, music notation, melody modeling, audio melody extraction

Contents

Preface

It has been more than 180 years since the American poet H. W. Longfellow wrote in his book *Outre-Mer: A Pilgrimage Beyond the Sea* often quoted and now well-known sentence:

> *Music is the universal language of mankind.*

He definitely was not the first nor the last one who had such ideas. Obviously, his thoughts behind this sentence were something slightly different at the beginning of Romantic era than any explicit relation between music and language. Comparison and search for similarities between music and language appear thorough the whole history of science and art. The most probable prehistorical vocal origin of music, from the non-linguistic utterances as expressing happiness or crying and weeping, just supports the proximity of such relation.

Importance of the language-music relation can be seen on how often music composers and theoreticians were interested in such question. A good example can be the Czech composer Leoš Janáček who at the beginning of the 20^{th} century studied melody of speech in order to use natural melody of utterances in his operas. For many years he collected "speech tunes" and rhythms of people speaking around him and then he used them while composing.

On the other hand, for example the Russian-born American composer Igor Stravinsky was convinced that music cannot be treated as a language at all. In his *Chronicle of My Life* he claimed that music is powerless to express anything. In *The Birth of Tragedy from the Spirit of Music* Friedrich Nietzsche said that music is totally incomparable with language too, but his reasons were exactly opposite. He believed that music has a really strong expressive power and that any words "cannot bring us one step closer to the deepest meaning of music".

It is mostly my personal intuition that tells me that studying the relation between music and language could be useful. My experience from learning music strongly reminds me experience from learning foreign languages and a way of thinking about musical composition, especially while performing a piece of music, is very similar to thinking about language.

Thomas Kuhn claimed in his *The Structure of Scientific Revolutions* that science seems to be so successful because it focuses only on problems – puzzles, requiring nothing more than enough invention and wit. Following this advice, I will put aside the big thoughts of previous paragraphs and I will try to at least slightly contribute to few puzzles concerning application of Natural Language Processing methods in music processing. Doing so, using all the rigorous and "dry" techniques I hope I will manage not to forget that music is an art at the first place – art with a strong proportion of Dionysian element – best described by quotation from Nietzsche's *The Birth of Tragedy from the Spirit of Music*:

> *Even under the influence of the narcotic draught, of which songs of all primitive men and peoples speak, or with the potent coming of spring that penetrates all nature with joy, these Dionysian emotions awake, and as they grow in intensity everything subjective vanishes into complete self-forgetfulness.*

Introduction

As previously mentioned in the preface a lot of people have been thinking about how similar language and music are. There exist many views on that problem – linguistic, musical, philosophical, the information theory one. The research in this field can take very different directions – an example can be a neurological study [1] showing that the human brain activity measured by magnetic resonance during a jazz improvisation when musicians react to each other is very similar to the brain activity during a spoken dialog. On the other side of spectrum of possible approaches, Dobrian in [2] sees as the most significant similarity that both for language and music, the western civilization developed sophisticated written forms and uses this observation as a basis for other ideas.

The written form meant in [2] was the western musical notation. Despite being very elaborated, it is never able to capture all performing aspects, and similarly to prosody in language, its interpretation is culturally dependent. The only parts of musical performances precisely captured in the notation are the pitches of individual notes and the rhythm, again with some exceptions. Usually, the dynamics and some other interpretative remarks, which can be hardly formalized, are mentioned in the score too. The notation is not the only way how to formally capture these features of music. Except the classical notation, punched tapes were used to program orchestrions and player pianos from the 19^{th} century, and the same principle was used a century later in MIDI files. Both of them can be vaguely called a programming languages for music playing machines.

In the rest of the thesis, we silently assume that music notation has a lot in common with the written language, but this could and also should be a topic for a deep discussion. Music is considered to be a semiotic system in the Saussurean sense and a lot of work has been done in this area since the second half of the 20^{th} century. At least this fact can somehow justify our assumption that notes can be treated similarly as graphemes in language.

The Natural Language Processing (NLP) focuses mainly on the written form of languages. In the last twenty years the statistical NLP methods became very popular in a large number of applications and outperformed the traditional knowledge-engineering-based methods in many tasks. In these days, methods based on the theory of probability and the information theory are used in various applications including speech recognition, information retrieval and machine translation.

The theoretical foundations of statistical NLP are largely based on the information theory. In some way it is natural to use the information theory while thinking about music as well. Meyer in [3] provides strong arguments for treating music like this, which were later in 1990s partially confirmed by experiments by Manzara et al. in [4].

Previous thoughts lead us to the main goal of this thesis, which is to explore possibilities of using statistical NLP methods in processing music and discuss areas where such methods can be useful. For the simplicity we focus only on the western music and only on its melody as the prominent perceptional feature which is understandable for everyone without necessity of having any musical education. A melody model using techniques from language modeling as an

experimental practical application of these thoughts was developed, and further used in the Audio Melody Extraction (AME) task.

A lot of work in this field has been done – from theoretical works from the "early time" of the information theory in 1950s to practical applications in recent time. Deeper theoretical insight and summary of previous work in this field is provided in the chapter 1.

The following chapter describes ways of extracting symbolic melody from MIDI files. Language modeling techniques are used to create a melody model and further methods of processing the symbolic melodies data are proposed.

In the last chapter a practical usage of the model on an AME task is shown. Performance of the system is compared with the state-of-the-art techniques.

1. Related Work

In this theoretical introduction we focus on the linguistic and information theory views on music and the western music notation. These directions of thinking about music are the most important for this thesis. We do not include any Digital Signal Processing (DSP) theory in this chapter and focus only on the abstract view on music and its symbolic notation. The DSP is briefly discussed in Chapter 3.

1.1 More Detailed Theoretical Insight

There are some apparent analogies between linguistic theories and musicological theories. Currently widely used tonal theory of music, developed by a German musicologist Heinrich Schenker at the beginning of the 20$^{\text{th}}$ century, uses a system of transformational rules very similar to those which introduced Noam Chomsky in his theory of generative grammar in 1957. A deeper analogy between the generative grammar and the tonal music theory was developed later by Lerdahl and Jackendoff in [5], where the tonal musical theory is fully formalized. Despite being Chomsky's student, Jackendoff was skeptical about all Chomsky's linguistic theories considering them to be too "syntactocentric", and in linguistics sympathized more with the cognitive semantics school. The cognitive semantics is a part of the Cognitive Linguistics Movement. It rejects the traditional separation of linguistics into phonology, syntax, pragmatics, etc. and claims that the deepest underlying layer for utterances is the deep semantics from which everything is generated. These opinions are reflected also in the Generative Tonal Music Theory, where the authors focus more on describing the mental representation of music than on grammar, which is treated just as a consequence of the deeper levels.

A research about composer recognition [6] brought a table summarizing the analogies between particular levels of NLP and music processing (see Table 1.1). This distinction of levels is still a little bit ambiguous and can make an impression of being inconsistent in some cases. For example Bod in below mentioned research [7] about parsing melodies into phrases (see Section 1.4) works on a syntactical level according to this table, the same as the melody model we introduce in Chapter 2 which stays much more on surface.

	text processing	**music processing**
phonetics	recorded voice	recording
phonology	phonemes of the language	separated notes
morphology	word structure	notes in the score
syntax	word order	n-grams, note order
semantics	word meaning, POS	harmonic functions
pragmatics	meaning of sentences	musical phrase structure
discourse	context of a text	interpretation of a piece

Table 1.1: Levels of NLP and music processing (copied from [6])

Different mapping between the linguistic layers and music is brought by the famous conductor Leonard Bernstein in his series of lectures [8] from 1970s. He considers the notes to be phonemes, motifs to be morphemes, musical phrases words, sections clauses and movements to be sentences. Besides that he suggests a hypothesis that the melodic motifs can play the role of subjects whereas the accompanying harmony the role of predicates. He stresses the vocal origin of music which gives the music and language a common grounding. He sees s the biggest difference that languages can have both the communication and aesthetic function whereas the music has only the aesthetic function and is therefore a pure poetry. He considered the poetry itself to be a super-surface layer of the language generated by a set of transformation rules from the usual surface layer – the prose. The situation with music is different in this sense because the only observable layer is the super-surface layer and the deeper layers remain hidden for listeners.

In many ways similar ideas were experimentally confirmed in [9]. In this study n-grams model is created from pieces parsed into elementary motifs. Listeners are supposed to have "stored" the motifs in a lexicon the same way as words of language. These motifs then create the musical phrases which are claimed to be equivalents of sentences. This approach can be seen as a reflection of a more lexical approach in formal linguistics. A very detailed lexicon of motifs was manually created for this purpose but unfortunately is not publicly available.

A big difference between linguistics and musicology, mentioned both in [8] and [2] is that musicology has bigger influence on the music than linguistics on languages. Composers use the theoretical findings in their work and the theoreticians use existing pieces as a source of their research. This is so apparent that Dobrian in [2] calls that situation an egg-chicken confusion. Similar phenomena appear in languages just in exceptional situations as was the time of the Czech National Revival movement activity in 19^{th} century for the Czech language, or 19^{th} century in Norway where similar situation led to current existence of diglossia.

It was already mentioned that the capability to transfer meaning and its explicit use in communication is a typical language property. According to [2], music can certainly have a non-musical meaning, usually expressed by clichés, but mostly it has just the musical meaning, sometimes called the implicit meaning. Having said this we should clarify what the meaning in music is. In contrast to language, music lacs functional semantics. Of course, sometimes music is supposed to express some non-musical content, but it is usually limited to onomatopoetic-like "pictures" or invoking a particular mood.[16]

There have been several ways to approach music semantics since the half of the 20^{th} century, usually corresponding with what was the mainstream paradigm in the language semantics. The paradigm shift can be illustrated on examples of two characteristic papers, first from 1980 [10], second from 2010 [11], both of them claiming they were a novel approach to music semantics and that they build upon the state-of-art techniques in NLP.

In the first one, usage of Conceptual Dependency representation is suggested for music. In such representation, sentence "Mary took John a book" is written as: `(PTRANS (ACTOR MARY) (TO JOHN) (OBJECT BOOK))`. For the case of music semantics a set of semantic primitives was designed. It contains harmonic functions and their modificationsca, information which note from the melody is

rhythmically accented etc. An implementation of such system is discussed, but was probably never done.

In the newer one of the mentioned papers, the WordNet semantic network was used to create an ontology of tags users assigned to songs and their parts on web services like Last.fm. Then various machine learning classifiers were trained to tag unseen songs to provide the high-level semantic information – e.g. where is the refrain of the song, to what style does it belong, what is the mood of the song, where are the dramatic moments etc.

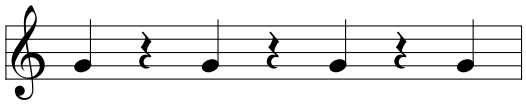
Having seen examples of how music semantics can be approached we can now come up with more abstract views on musical semantics. Very influential, almost exclusively information theory based approach to musical semantics is introduced by Meyer in [3]. It is a theoretical work with aesthetic and philosophical background. Despite this view was suppressed by more Chomskian-like approach for almost three decades, recently it became frequently cited. Meyer thinks that musical meaning manifests itself in confrontation of the listener expectation and perceived reality, which leads to a straightforward use of information theory. He uses the following definition:

> *Musical meaning arises when an antecedent situation, requiring an estimate as to the probable modes of pattern continuation, produces uncertainty as to the temporal-tonal nature of the expected consequent.*

Three stages of developing a meaning by the listener are distinguished. The *hypothetical meaning* is derived basically just by admitting information. Just a deviation in an expected consequence is perceived – something less probable, but still possible in the particular style. The *evident meaning* is based on a mental feedback, it provides backward understanding of what just happened. We can say it is a result of finding that the situation was actually much higher in entropy than it was originally thought to be. Final stage of perceiving musical meaning is deriving the *determinate meaning*. It means understanding music on the architectonic level – different themes and phrases are recognized and given into a broader context.

From this point of view, music can be modeled as a Markov process. It is important to mention that not a simple n-gram based model as is used in this thesis, but a much more complex model is meant, which can capture also the determinate meaning. The big information redundancy in music is caused by the non free choice of symbols and is therefore similar to what we can observe in language. In language, grammar rules are the reasons for this, whereas in music cultural rules play important role. Meyer calls this redundancy "cultural noise".

Being familiar with Meyer's conclusions and some later practical applications using information theory (some of them are mentioned in 1.3), Dobrian in [13] does not support information theory view on music. He claims that the key to understand music is not in trying to find structures in the music itself, but to properly model the listeners perception which according to him cannot be reduced to quantification of listeners expectation in form of probability distributions. He illustrates this idea on a thought experiment. When people come to a piano concert and hear a following beginning of the piece:


,

they would probably expect to hear the same note again. Considering the Occam's razor to be a natural way of critical thinking, Dobrian concludes there can hardly exist a cognitive model based on expectation not predicting the same note again. The whole piece would then be expected to consist just of separated g' notes, which would certainly be a false expectation.

This point of view, probably influenced by a general skepticism to quantitative methods in social sciences and humanities starting in 1990s, underestimates possibilities of how a statistical model can look like. Still, it indicates possible limitations of information theory approach, especially the problem of very simplistic models. He may be right in the sense that Markov processes in general are not strong enough to capture also the determinate meaning in music, similarly as the language modeling tools fail to capture long distance dependencies within sentences or even within whole texts. Anyway, developing musical meaning from expectation can be considered to be a valid paradigm now in all NLP-like musical applications.

Further arguments for information theory usefulness are presented by Manzara's and Witten's experiment in [4]. It is a music remake of the famous experiments Claude Shannon made at the end of 1940s and in early 1950s with written English.[12]

Shannon's goal at that time was to compute information redundancy of printed English. In the experiment people were asked to try to guess a letter following in a text while seeing all the previous letters, until they guessed right. The perceptional entropy of English language was estimated from that data.

Melodies of Bach Chorales were used for the music version of this experiment. People who participated in the experiment were trained musicians, all of them familiar with Bach's music. The musicians were chosen to avoid higher entropy caused by surprise of participants who were not familiar with the style of the music. Harmony was totally eliminated and rhythm was known beforehand, so the only feature being guessed was the pitch of the melody notes. Results show that an information theory approach similar to the language one can be easily applied and can help to characterize the short-term structure of music.

A computational model for the Bach's chorales in [14], which was a linear combination of n-gram models using various ways of parsing the scores, led to a little bit higher entropy. The observed difference between the human cognitive model and computational model is comparable with the Shannon experiment with English. Using n-grams statistics, the hypothetical and partially also the evident meaning from the Meyer's hierarchy can be captured. This shows computational models do not have to be much complex to provide satisfiable results. The model in [6] using music parsed by beats which is based only on simple n-grams gives also good results. In that study it was experimentally discovered that best results are obtained with 7-grams, which is history of approximately 2 bars, which corresponds to Meyer's estimate of human memory in deriving hypothetical meaning.

Another interesting fact considering the similarity between music and language was observed in several studies. The distribution of musical phrases in classical pieces (see [6]), chords in popular music (see [15]) and also individual

notes (with a very detailed feature set) in most of the classical pieces for piano and organ [16] satisfies the Zipf-Mandelbrot law.

1.2 The Role of Melody

Because in this thesis we work with melody extracted from the polyphonic scores we should also discuss the role of melody from a more theoretical point of view.

According to the The New Grove Dictionary of Music the melody has a following definition [17]:

> *Melody, defined as pitched sounds arranged in musical time in accordance with given cultural conventions and constraints, represents a universal human phenomenon traceable to prehistoric times.*

Even though the melody is one of the fundamental aspects of the music composition, sometimes it is difficult to say what exactly the melody is, even for the human listeners. In more complicated compositions (as symphonies or other big classical pieces) the melodic motifs often overlap and in some parts harmony plays much more important role than the melody. On the other hand, in case of simpler compositions (folk music, popular music, simpler classical pieces), it is the melody what is most important for listeners and it is almost unambiguous.

Melody is one of the simplest features of music, relatively clearly defined and understandable for everyone. This may be one of the reasons why melody is very often a subject of computational analyses of music. Because of their theoretical knowledge and cultural experience trained musicians usually do not have problems to assign a suitable harmony to any melody. Assuming the knowledge of harmonic rules, harmony must be somehow encoded in the melody, even though not entirely. We can think of melody extraction as of a loss compression of a piece. All the further processing of the melody of a piece means some loss of information which is unfortunately often vital for the listeners music experience. No matter how interesting the conclusions of melody analysis can be, it is unlikely to bring us closer to understanding human perception of music which is probably the hidden ultimate goal of any computational analysis of music.

In [18] Umberto Eco blames the mass culture for reducing very complicated pieces of music, like classical symphonies, to simple melodies one can whistle. Making this "brutal" simplification a lot of the composers' intentions is totally ignored. On the other hand, melody is still a fundamental aspect of music. The possibility to extract and represent it quite easily, makes it a perfect object of the computational analysis. Furthermore, even in case of complicated compositions, melody is something that one can sing to each other in order to refer to a particular part of a composition or that can be used as an identification sign in music retrieval system. Therefore studying it is useful also from the practical point of view.

1.3 History of Computational Music Processing

Development of paradigms in the music processing during its history follows the paradigm development in computational linguistics and in other artificial-

intelligence-like fields of study. After some initial experiments with the information theory in 1950s and 1960s, simple systems using the hand-written rules appeared. Researchers mostly preferred the knowledge-engineering approach which led to hardly any generally and easily applicable results. The increase of availability of computational capacity allowed statistic-based systems to grow in popularity from 1990s till nowadays.

Pearce in [19] mentions a work of Lejaren Hiller and his colleagues from 1960s. They made a deep information theory analysis of a quite restricted domain of music compositions – Mozarts' and Beethovens' sonatas and one Anton Webern's symphony – and found significant regularities. Based on that they were able to infer a lot of mathematically formalized rules that apply in these compositions. Few years before that Hiller together with Leonard Isaacson had created a software using textbook rules of harmony, voice leading, and style which imitated pieces of German composers from 19^{th} century. The authors described the results as boring pieces which sounded like having been composed by a forgotten German country composer. Soon after that another algorithmic composition projects came (see [20]), usually working on the same principle.

These relatively good results led to optimism in 1960s and 1970s when researches believed that the music processing tasks could not be so difficult and that at least some tasks like style recognition could have soon achieved a good solution. A lot of domain restricted prototypes were developed at that time. The paper [21] shows how demanding developments of such a system can be on an example of a magnificent work by David Cope who has patiently derived grammars and logics for different styles of composition and using them he was able to generate music in the style of Bach, Beethoven, Mahler and others. It took him more than 20 years to make the system work well. The final result was presented in 2001.

The development of the MIDI format which first appeared in 1983 was important moment for the music processing. It is a binary format originally designed for the instrument interconnection, later used also for storing the music. It contains the same kind of information as the classical notation does – which instrument plays, times of starts and ends of notes, their loudness, current position of the sustain pedal etc. The format captures a life performance for further processing. It became the dominant standard for storing symbolically written music. As a consequence of that a lot of data of all music styles is now available in this format.

At the end of 1980s and in 1990s the neural networks raised in popularity in the artificial intelligence fields of studies. Some music related research papers using neural networks also appeared. [27] [28]

In the recent research almost exclusively statistical methods are used. Sometimes we can find the term *empirical induction methods* used to stress the fact that all knowledge is inferred just from data.

Except the n-gram approach which is discussed in a more detailed way in the next section, dictionary based approach appears in some applications. First a dictionary of possible melody motifs is used to find the most probable parsing of the melody and then these motifs are treated as states of a Markov chain. Dubnov et al. in [9] use this technique successfully for the style recognition on MIDI files in a wide range of pieces from early renaissance and baroque music to hard-bop jazz.

Hidden Markov Models are used in programs *M* and *Jam Factory* by David Zicarelli from 1987 [29] which generate jazz solos based on the MIDI input they receive. Big progress in this area was brought by an experimental software *The Continuator* [30] developed by Sony Computer Science Laboratory in Paris in 2002. The software continues with the music it received via a MIDI sequencer in a real time. Very efficient pattern recognition mechanism based on prefix trees was designed for analysis of the input in real time. On its webpage[1] there is also a short information about this program passing a "music version of the Turing Test". People taking part in the experiment were asked to determine which performances are completely done by humans and which were from a certain moment machine generated. The result was that the number of correct answers was not significantly higher than the number of false answers.

Seeing all of the history from todays perspective Cont in [21] thinks that the biggest problem of all these applications was that they were very domain restricted and did not cover the usual way of humans thinking about music. By that she meant mostly switching the cognitive model according to larger context, which actually means changing the perceived style. In overcoming this problem of switching models she sees the crucial step in further development of music processing. Her conclusions also correspond with Meyer's idea that just simple studying of n-grams cannot describe the whole structure of a piece of music and that different musical styles are different "languages" where different rules hold.

1.4 An Overview of Recent Work

In this section we will describe recent applications of the NLP-like methods in music processing. The mentioned tasks cover most of the music processing problems being currently solved. The tasks are provided with an example solution on which the principles are shown, but an exhaustive overview of state of the art methods is not provided. It is typical for most of the mentioned works that they do not use explicitly any of the linguistic formalisms and usually use n-gram statics in a similar manner as they are used in NLP.

From the view of the classical linguistics which considers all morphology, syntax and semantics, it is still a little bit surprising how well can n-gram models describe the language. This gives us a strong hope that in music, where an immediate expectation plays an important role in the human perception, the n-gram model could be successful as well.

One of the most interesting uses of linguistic methods is Rens Bod's work [7] about creating probabilistic grammars for melodies of folk songs based on the Essen Folksong Collection.

The Essen Folksong Collection [22] is a corpus of melodies of mostly European folk songs. Except other information (origin, lyrics etc.) it contains hierarchical structures of phrases, because natural parsing of melodies into phrases differs among different nations. Basically it is a treebank for folk song. Melody is represented relatively to the tonality as *do*, *re*, *mi* etc. syllables.

Bod tried various ways of creating a probabilistic grammar for folk song phrases trained on the collection. Using an extended version of Markov Grammar

[1] http://csl.snoy.fr/~pachet/Continuator

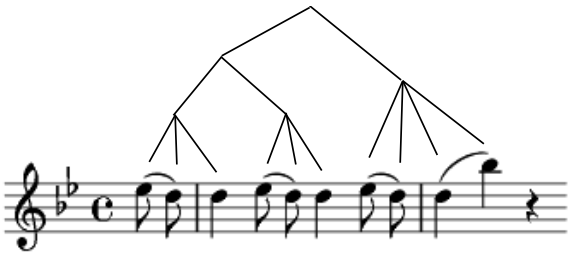


Figure 1.1: An example of phrases parsing of the beginning of the Mozart's G minor Symphony

Technique he achieved precision of 86 % on the test data randomly chosen from the collection.

In this work we can see a similar approach as in a work of Dubnov [9], but Bod could avoid using the dictionary of motifs and use just n-grams because he had additional information from the phrase structure annotation.

Bod considers the notes to be equivalents of graphemes or phonemes and tries to infer syntactic rules for them. What is called syntax in this paper is in fact something as a generative morphology gradually changing to a generative syntax. Moreover, these grammars were build on the folk music where assigning the harmony to the melody can be done almost unambiguously. The trained grammars in fact use the harmonic and melodic information simultaneously. Anyway, this seems to be the only study which attempted to use machine learning to build some music grammars. These can be utilized in the AME.

In [6] n-gram statistics are used for composer recognition task. A table of melodic and rhythmic trigrams called a "profile" is computed for each piece and compared with reference profiles compiled from sets of pieces of particular styles. The style having the closest profile to the examined piece's profile is then assigned to the piece. Despite being relatively simple, the method achieved very good results. The paper also contains a detailed theoretical introduction about using analogies between NLP and music processing.

In recent years a big progress in a lot of music processing methods has been done thanks to the annual contest Music Information Retrieval EXchange (MIREX). The contest covers a large spectrum of both symbolic processing and DSP tasks, in particular there are: audio artist identification, audio chord detection, audio classical composer identification, audio genre classification, audio melody extraction, audio music mood classification, query-by-singing/humming, query-by-tapping, real-time audio to score alignment and symbolic melodic similarity. In two other symbolic processing tasks and symbolic genre classification, symbolic key detection, no one participated in last seven years.

However, these two problems were mostly approached as a machine learning problem where features like interval frequency, frequency of syncopations were used to train various machine learning classifiers.

Algorithms for symbolic genre classification often use n-gram statistics as in case of [6]. Almost the same method was used for example in [23]. More information retrieval methods and similarity metrics were explored in [24], submission by authors of the previous study [6]. A hierarchical index based on n-grams is

shown in [25]. The most successful algorithm [26] and many others use a totally different approach. The melodies are interpreted geometrically as curves in a two-dimensional plain and by comparing curves produced by this transformation the genre to which the piece belong is determined.

1.5 Summary

Unfortunately, in this theses we did not manage to do a theoretical research which would be deep enough to firmly say what exactly the relationship between language and music is. The history of computational processing of music gives us hope there are some deep theoretical connections giving raise to possibility of successful applications of linguistic tools as grammars or n-gram statics in music processing.

Both fields also experienced very similar development of scientific paradigms in the second half of the 20^{th} century. Statistical methods now applied in computational linguistic are much more likely to be applied in music processing than earlier methods relying heavily on the linguistic formalism. This may lead us to unsubstantiated speculations that using statistical methods may bring us closer to revealing some very basic common features of music and language. Maybe only the arbitrarily chosen scientific views of linguists and musicologists on their fields of study prevent us to see such connections directly.

2. Symbolic Melody Processing

The aim of this chapter is to describe and discuss extracting melody from polyphonic symbolic data and further processing of such melodies using techniques common in NLP.

Unfortunately, there is no general corpus of raw melodies available except the Essen Folksong Collection [22] (described in 1.4), which focuses on folk songs. A useful dataset was created for purposes of [31]. It is a training corpus for melody track identification consisting of more than 700 hours of classical, popular and jazz music. However, only the potentially polyphonic tracks containing the melody are annotated, not the melodies themselves. Besides that, a big amount of unannotated data is freely available on the Internet. Melody extraction from such data is described in the next section.

On the extracted melodies various NLP method can be applied, among them language modeling as the most straightforward use of the data. In Section 2.2 building of a statistical melody model from the data by the language modeling techniques is described and evaluated.

Another possibilities, unfortunately remaining beyond the scope of this thesis, are unsupervised "word" segmentation and parsing which can be used to create grammar rules as in [7] possibly applicable in the audio melody extraction. Semiotic and musicological analysis of all previously mentioned methods can be also very interesting. These ideas are briefly described in Section 2.3 despite not being supported by any experiments.

2.1 Melody Extraction

Because of the lack of the pure melody data, the first step in preprocessing of the symbolic melody data must be the melody extraction from the MIDI files. By extracting the melody we mean computing a sequence of consecutive notes represented just by their pitch and duration, which is supposed to be as close as possible to what humans perceive as a melody. Interpretative aspects of the performances such as dynamics and tempo changes are neglected as well as all the music background information contained in the harmony.

2.1.1 State of the Art

The melody extraction from polyphony is an important task in Music Information Retrieval (MIR) and several algorithms have been developed. Widely used skyline algorithm (introduced in 1998 [32]) follows the simple psychoacoustic fact that the highest pitched note in the polyphony is usually a part of the perceived melody. The simplest implementation of the algorithm leads to frequent changes of pitches in the estimated melody which have very disturbing effects for the listeners. Later some heuristics to get rid of the negative properties of the algorithm appeared. The skyline algorithm also fails on complicated orchestral music, where violins often play the highest pitches despite the main melody being played by lower pitched, but louder instruments. An example of an algorithm result for a simple piano score is in Figure 2.1.

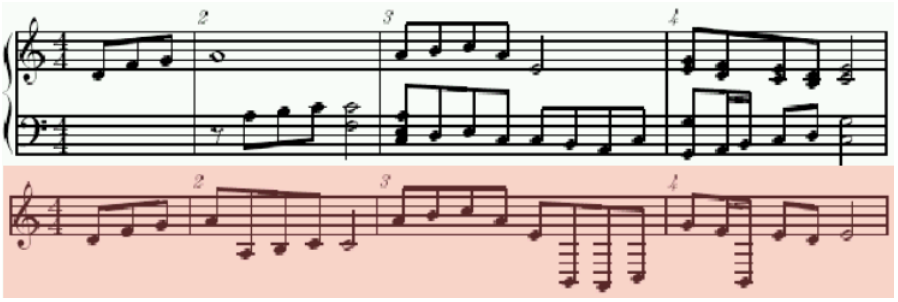


Figure 2.1: An Example of the algorithm from [32] applied on simple piano score. Figure copied from [34]

Another approach is selecting a track containing the melody most likely. Some algorithms are based on the assumption that the track with the melody is the loudest one or that it has the highest entropy. This assumes that the melody is richer in information than the accompanying instruments. Anyway, this approaches fails if the melody is split into more tracks. Later, more complex algorithms combining properties of these two approaches appeared. One of the most complex ones [33] uses the agglomerative clustering on pitch histograms to group similar tracks. From each track cluster a track with the highest entropy is chosen and a modification of the skyline algorithm is applied just on the selected tracks. Despite the declared accuracy of more than 90 %, the empirical evaluation of our implementation of this algorithm on our dataset gave very poor results. An example of applying this algorithm on the same score as the previous one is in Figure 2.2. Only the modification of the skyline algorithm is apparent in the figure.

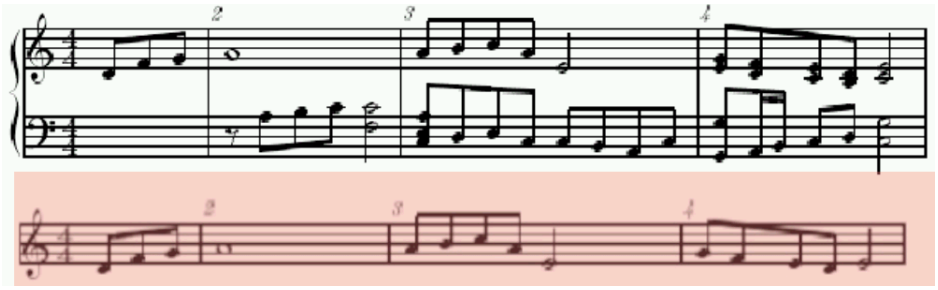


Figure 2.2: An example of the algorithm from [33] on the same simple piano score as in Figure 2.1 copied from [34].

The algorithm [31] which reported the highest accuracy in selecting the melody track is based on supervised machine learning techniques. The training data for the algorithm consist of 3,000 MIDI files downloaded from the Internet. Tracks in these files were manually annotated whether they contain the melody or not. The corpus consists of 331 files of classical music, 1,223 of jazz music and 1,360 karaoke accompaniment files of popular music.

From each track a set of features described in Table 2.1 is extracted. For all the features, except those in the category of track information, there is also a normalized form where the normalized value of feature v in track i in file F is

category	features
track information	normalized duration duration relative to other tracks in the file number of notes relative duration of non silence relative duration of polyphony moments
pitch	highest lowest mean standard deviation
pitch intervals	number of different intervals largest smallest mean mode standard deviation
note durations	longest shortest mean standard deviation

Table 2.1: Overview of features used for the melody track identification, taken from [31]

$$v_i^{(norm)} = \frac{v_i - \min_{j \in F} v_j}{\max_{j \in F} v_j - \min_{j \in F} v_j}.$$

From each style 200 files were chosen as training data and the rest of them as test data. The training data consisted of the most unambiguous cases according to the annotators. The random forest classifier turned to be the best performing one for this task (probably because the feature set is quite large and the chosen method does not suffer from the curse of dimensionality). It also showed up that the classifier gives the best performance when it all the train data joined together. The paper reports the average successful melody track identification percentage of 81.2 %.

2.1.2 The Used Algorithm

Thousands of MIDI files from Classical Piano Midi Project, The Classical MIDI collection, web page of Music Files ltd., server Free MIDI and the Mutopia project were downloaded. See Table 2.2 for overviews of the files by styles. To these 556 hours of raw MIDI files, 161 hours of MIDI files from the corpus with annotated melody tracks from [31] were added.

Previous empirical evaluation of the result of the skyline algorithm with some added heuristics showed that such algorithm can be successfully used only for files which are relatively simple. After an unsuccessful experiment with [33], method from [31] was used to select the melody tracks.

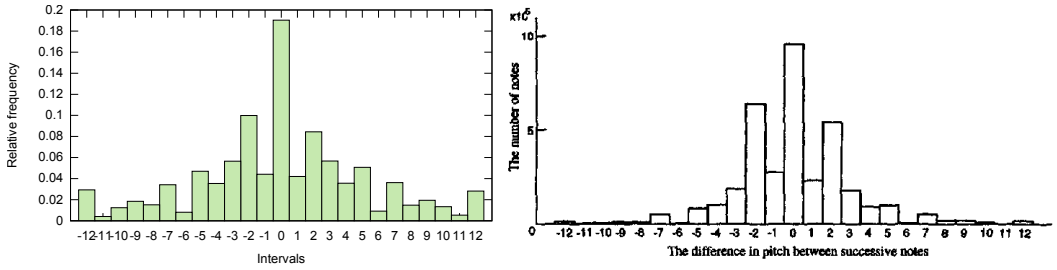


Figure 2.3: Interval histograms of melodies from popular music songs in our corpus (on the left) and interval histogram copied from [36] (on the right)

The skyline algorithm is then applied on the tracks either classified as melodic by the classifier or manually annotated in the corpus. Despite the reported success rate of 81.2 % in detecting melody tracks in MIDI files, the classifier was able to find the melody only in 63 % of files with classical music and in 47 % of files with popular music in our corpus. On the other hand, while listening to the results it seemed that the tracks were selected very well. The algorithm probably suffers from low recall, while having relatively high precision. Because the classifier captures the melodicity of tracks very well, we allow the classifier to select more tracks from a single file as the melodic ones. In the classical music files the tracks often alternate at containing the melody, therefore it is relatively frequent phenomenon that more tracks are classified as melodic.

The psychoacoustics based heuristic from [35] is used. Notes shorter than 0.02 s are removed and notes longer than 2 s are shortened to this length. As another heuristic all notes having the pitch lower than small g are removed.

Then the skyline algorithm is performed in the way it was used in Kosugi's MIR system [36]. It means that notes which are at least at one moment the highest in the score are selected to be a part of the melody. Then the notes are shortened to end at the latest at the moment the next selected note starts regardless of which one is higher.

The melody tracks may contain a lot of silence. Therefore the long periods of silence are cut out from the tracks. That splits the melodies into more melody snippets. Only snippets containing at least 8 notes are kept.

From 10,420 MIDI files of total length 716:52:16, 22,612 melody snippets were extracted of total length of 408:03:35. An overview of the amount of available data sorted by genres is presented in Table 2.2.

It is difficult to evaluate the performance of the algorithm precisely without having painstakingly created test data. Listening to randomly picked melodies revealed that the extracted melodies might not be exactly what listeners perceive as the melodies in the particular pieces. Still, the results sound always very melodically and fit to the styles and composers they belong to.

Another argument to support the opinion that the algorithm performs reasonably well may be the comparison of the interval histogram with the one presented in [36]. The paper depicts the histogram of intervals and reports it to look exactly as expected. Unfortunately, the exact figures from [36] are not available, but the visual comparison of the plots in Figure 2.3 suggests that the interval distributions are similar.

	all files	extracted melody
Renaissance	11:18:28 404 files	12:49:21 2 209 snippets
Baroque	97:33:32 2 502 files	91:09:59 5 353 snippets
Classicism	86:18:06 810 files	48:40:30 2 636 snippets
Romanticism	143:59:13 1 484 files	59:58:06 3 092 snippets
Impressionism	10:04:35 164 files	5:02:18 347 snippets
20$^{\text{th}}$ century	83:50:58 1 003 files	34:54:55 2 209 snippets
Classical music	434:04:52 6 366 files	252:35:09 14 359 snippets
Jazz	44:42:42 837 files	38:11:10 1 308
Rock	83:15:29 1 453 files	32:33:41 2 380 snippets
Pop	37:58:12 712 files	18:32:56 1 004 snippets
Popular music	121:13:41 2 165 files	51:06:37 3 384 snippets
Training data from [31]	117:36:07 1 052 files	66:10:39 3 561 snippets
Total	716:52:16 10 420 files	408:03:35 22 612 snippets

Table 2.2: Overview of the data amount available for the computations

2.2 Melody Modeling

2.2.1 Language Modeling Overview

A language model is a statistical model that assigns a probability to a sequence of m words $\mathsf{P}(w_1, \ldots, w_m)$ which is supposed not to differ much from the probability of the word sequence being uttered in a given language.

Texts are usually approximated as an n-th order Markov chain where each word probability is conditioned only on the previous $n-1$ words. For a sequence of m words we can formally write:

$$\mathsf{P}(w_1, \ldots, w_m) = \prod_{i=1}^{m} \mathsf{P}(w_i | w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} \mathsf{P}(w_i | w_{i-(n-1)}, \ldots, w_{i-1}).$$

The n-gram probabilities are estimated from a training corpus using the maximum likelihood estimate. Because it often happens that a lot of n-grams which

are plausible in the language were not observed in the training data, it is necessary to modify the estimated probabilities in order to prevent the model from assigning zero probabilities to sentences containing unobserved n-grams.

Generally, there are two approaches to avoid this problem. These are discounting – which means redistributing a part of the probability mass to the unobserved n-grams – and interpolation – using a linear combination of higher and lower order models to be able to cover all the possible n-grams.

There are various methods both for interpolation and discounting. Here, we will explain the Good-Turing discounting method which is used in the melody model introduced later.

Estimated frequency F of n-gram $w_i, \dots w_{i+n-1}$ is defined as

$$F_X = \frac{c(w_1, \dots w_{i+n-1})}{c(w_1, \dots, w_{i+n-2})} \cdot \frac{E(c(w_1, \dots w_{i+n-1})) + 1}{E(c(w_1, \dots w_{i+n-1}))},$$

where $c(x)$ is the count of n-gram x and $E(y)$ is a number of n-grams that appear exactly y times in the training corpus.

A more detailed overview of language modeling techniques can be found in [38].

Except for the methods based on counting n-gram occurrences in the training corpus, neural networks can be used for language modeling. Better availability of computational capacity recently allowed researchers to train deep neural networks with many hidden layers. In [39] a neural network language model is trained and performs slightly better than an 4-gram language model.

To model the melody development not only methods from NLP can be applied. A computational model of melody [40] has been created based purely on the hand-written formulas. Except for the pitch and rhythm it also uses other features such the as local pitch variance, the key of the piece or accompanying harmony. The formulas are based both on cognitive science results and author's introspection. Most of the parameters were estimated from data or from known psychoacoustics facts.

2.2.2 Melody Preprocessing

The corpus of melodies represented as a sequence of notes' pitches and durations would be extremely sparse and hardly any sequence of notes would appear more than once in the data. Therefore we represent the melody similarly as in [23] and as in the music visualization system MIDVIS [37] – as a sequence of differences between the consecutive notes. This representation also somehow captures equality of melodies as humans understand it – when a melody is shifted in pitch, slowed-down or accelerated, people usually recognize it as being the same one. The differences between notes are represented as pitch interval as a number of semitones from the previous note or string "`pause`" in case of pauses respectively, and a ratio between the lengths of the consecutive notes. Various methods of encode the duration changes were tested – in particular various precisions of rounding the ratios and their binary logarithms. To be able to capture the shortening of notes as accurately as their prolonging while using a plain ratio the following notation was used:

$$toString(d_1, d_2) = \begin{cases} \text{round}(d_2/d_1) & d_2 \geq d_1 \\ \text{"1/"} + \text{round}(d_1/d_2) & \text{otherwise} \end{cases}, \tag{2.1}$$

where d_1 and d_2 are durations of two adjacent notes. Both recently mentioned papers use rounded binary logarithm to encode the duration change, using the assumption that the note durations most frequently differ by powers of two. While using the 2.1 method, melody models achieved lower perplexity than while using the rounded binary logarithm of the ratios. These computations, described in more detail in Section 2.2.3, showed the most suitable way of encoding the duration changes is 2.1 rounded to integers.

The previous theoretical discussion about music and language leads us to a curiosity whether the note changes as units of melody, extracted in the described way, satisfy the Zipf-Mandelbrot's law as the lexical units in languages do [38]. Moreover, exponential dropping of word frequencies is an important assumption used in various language modeling techniques.

If the Zipf-Mandelbrot's law holds, the frequencies f of tokens and their position r in the ordered list of the frequencies satisfies equation:

$$f \approx P(r + \rho)^{-B}$$

for some $P, \rho > 0$ and $B \geq 1$.

For all particular styles the best-fitting Zipfian curves were computed using the linear regression on 85 % of randomly chosen data. The square error of the rest 15 % of data fitting the curves was tested to be unequal to zero using the t-test.

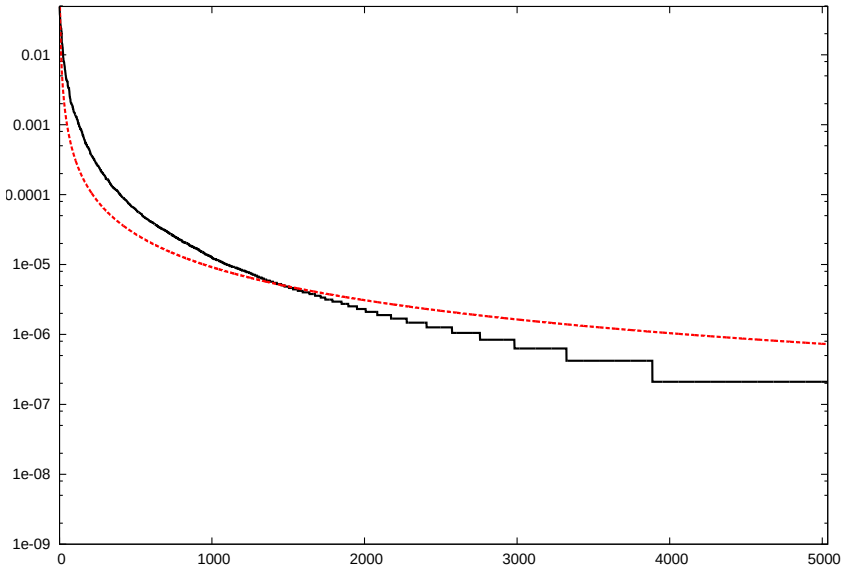


Figure 2.4: Ordered distribution of notes in the whole training corpus (black) with the best-fitting Zipfian curve (red) with logarithmically scaled relative frequency. The values of parameters are: $P = 0.47$, $\rho = 0.24$, $B = 1.57$.

The computations showed that the Zipf'-Mandelbrot's law does not hold for any of the styles, except for the whole corpus together. The plots in Attachment A.1 show that the distribution of notes in other styles is very close to the Zipfian. The plot of the Zipfian curve fitting the whole corpus note distribution is in Figure 2.4. The fact that the law does not hold for the individual styles may be caused by too many rarely occurring tokens. Such tokens then have similar relative frequency even though their real music probability differs more significantly.

Using the NLP terminology we call the set of tokens observed in the training data the vocabulary. No matter which melody representation we choose, there still will be some out-of-vocabulary tokens in the test data and a lot of others out-of-vocabulary tokens could occur during actual use of the model in an AME system where also a lot of improbable hypotheses must be evaluated. The reason for that is that the theoretically possible vocabulary set is very large. It includes all possible combinations of pitch intervals and discretized duration ratios. Choosing the precision of the duration ratios is in fact a very simple method of clustering the similar tokens. Additional clustering methods could be used to ensure that the whole vocabulary is covered. We can understand it as an analogy of creating hand-crafted word classes based on rules which were used for example in the named entity recognizer in [41].

To eliminate out-of-vocabulary tokens we choose following heuristic clustering method. A duration ratio $d \geq 1$ is substituted by $c_d(d)$, ratio in form of "$1/d$" is substituted by "$1/c_d(d)$" where

$$c_d(d) = \begin{cases} d & \text{if } d \leq 8 \\ \text{"}\texttt{9to16}\text{"} & \text{if } d > 8 \;\&\; d \leq 16 \\ \text{"}\texttt{17to32}\text{"} & \text{if } d > 16 \;\&\; d \leq 32 \\ \text{"}\texttt{over32}\text{"} & \text{if } d > 32 \end{cases} .$$

For the pitch interval heuristic clustering two methods were tested. In the first one intervals from -16 to 16 remain unchanged, bigger intervals are divided into groups of smaller than two octaves, bigger than two octaves but smaller than three octaves, and over three octaves, separately for both the positive and negative intervals. In the second method, intervals were substituted by their size modulo 12, again separately for the positive and negative intervals. Despite the last method significantly lowers the perplexity of models, it is not suitable to be used in an AME system because such model would not help with octave mismatches which are the most frequent errors in such systems.

2.2.3 Building the Model

Melody n-gram models of music of different styles were created based on the machine extracted melodies.

After first experiments with our own implementation in Perl, the SRILM toolkit [42] was used to make the computations faster. It is a toolkit for building and applying statistical language models, primarily for use in speech recognition, statistical tagging and segmentation, and machine translation, developed at the John Hopkins University.

The Good-Turing discounting was used. Unlike Witten-Bell and Kneser-Ney smoothing it does not use a set of back-off models and we can therefore expect the computation to be fast even for longer note sequences.

For each style 90 % of melody snippets was used as training data, the rest 10 % was used as test data. Overview of the data size is in Table 2.2.

Different n-grams length were tested. Detailed results are tabulated in A.2.1. As the n-gram length grows, perplexity of the test data drops. The last big drop in perplexity is between order 9 and 10. The most suitable n-gram order would therefore be 10.

Models for each of the styles were tested on a test set of the same style and also on test sets of all the other styles. It holds that the more training data is used, the better performance on all test sets is achieved. The only exception is the Renaissance music which is in many senses different from the rest of the styles. The model trained on Renaissance melodies beats all other models on the Renaissance test set, on the other hand it has very bad results on all the other sets. For all the other test sets the lowest perplexity was reached using the model trained on all available training data. Based on that experiment only the model trained on all data is used in the AME system.

This is also interesting from the theoretical point of view. Meyer in [3] claimed that while creating a model of human music perception, different styles must be treated separately because they are as different as natural languages are. Hardly any conclusion can be made from this result because our model does not include harmony. Moreover, it is possible that the results would be totally different if we had much more data from each style.

The models reach per token perplexity of more than twenty on each of the test sets of different styles. These numbers are higher than those that were computed in [14] in the experiments with the Bach chorale, but in those experiments the rhythm was already known and just pitches were estimated.

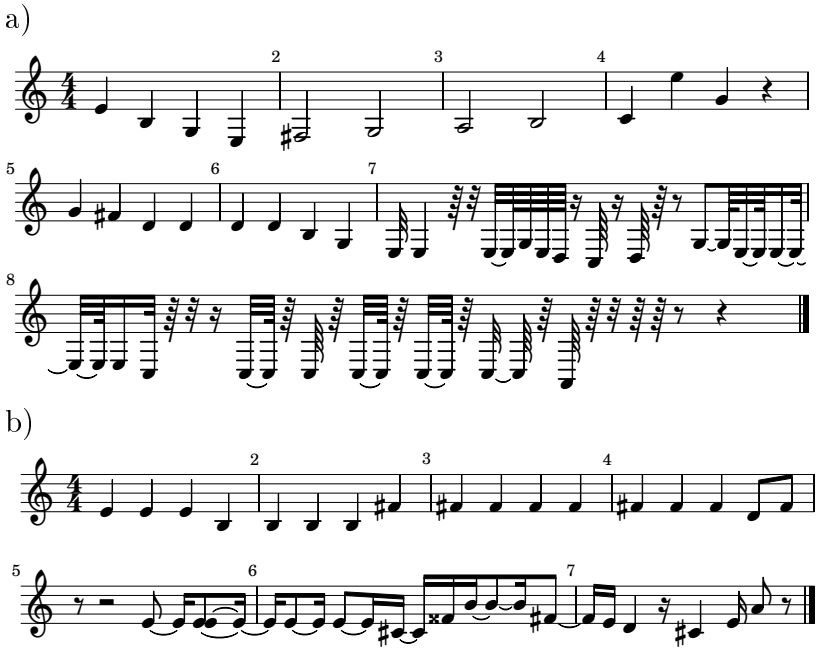


Figure 2.5: Examples of melodies generated by the language models

A random walk on the model can be used for melody generation. Examples of such melodies can be seen in Figure 2.5. Because the model uses only note changes, quarter note e was arbitrarily chosen as the melody start. Some phenomena which would be normally expressed using e.g. the staccato sign or music ornamentation are captured explicitly. This causes problems with the resulting notation readability. The melodies sounds locally plausible, but it can be easily recognized that they were not created by a human composer. Generally, we can say that the generated melodies tend to be in minor scales and in the 4-beat rhythm and do not sound naturally.

2.3 Other Options for Using NLP Techniques

A challenging problem we originally wanted to address in this thesis was what could be an analogy of a word or of a morpheme in the context of music. As a baseline solution it would be possible to re-implement the Morfessor [44], a tool for unsupervised morpheme segmentation originally from 1989. The method is based on optimization of a cost function that minimizes both information necessary to store a set of morphemes and information necessary to encode the training corpus using the set of morphemes. The cost function C is formally expressed as

$$C = \sum_{i \in morphemes} k \cdot length(m_i) + \sum_{j \in tokens} -\log_2 \left(\hat{p}(m_j)\right),$$

where k is a number of bits needed to encode one character of the alphabet and $\hat{p}(m_i)$ is the maximum likelihood estimate of the probability of the token m_i in the training corpus.

Trying all possible splits of the corpus would lead to exponential complexity. Originally a heuristic solution was used. Recently more efficient optimization methods based on sampling appeared.

The only experiment we did in this area was trying to use the "latticelm" tool based on Bayesian statistics [43] where the segmentation is modeled as the Chinese Restaurant Process and Gibbs sampling is used. Unfortunately, the tool was not able to converge to a solution on our data because of the large vocabulary size.

First, it would be necessary to test how much the received segmentation covers a test data. In case of success it would be interesting to compare such note sequences with what the musicologists call motifs or phrases and test another statistical NLP methods.

One of easily explorable methods is creating Brown's classes by agglomerative clustering of words based on mutual information. While used on language, these classes usually correspond to parts of speech and other morphological features of the words. In case of music motifs correspondence of the classes with musicological classification could be explored.

From the compositional perspective it would be meaningful to explore machine estimated collocations (sequences of tokens that occur together more frequently that they would by chance given their frequencies in the corpus). Unsupervised parsing could be used to estimate melody grammars. However, methods of unsupervised parsing still does not perform particularly well and chance success

of their application would be doubtful. Exploration of both methods mentioned above could lead to findings which compared with monographs about composition could give interesting theoretical results.

2.4 Summary

In this chapter some methods of symbolic melody processing were discussed. Various methods of symbolic melody extraction were tested. A method combining machine learned classifier for melody track selection and a set of psychoacoustics heuristics to receive the actual melody was used to extract melodies from a big corpus of MIDI files downloaded from the Internet. Although the results seem to be acceptable, they were not tested on a reliable test set and therefore there is a danger that there might be a systematic error which could influence further computations and which cannot be noticed by random proof listening to the melodies .

Consecutive notes changes were used to represent the melody to avoid data sparseness. We verified that the melodies represented this way satisfy the Zipf-Mandelbrot's law which was mentioned in many previous works. The fact that the melody model using the whole corpus performed best on all the test sets and that the Zipf-Mandelbrot's law reliably holds only for the whole corpus probably means that the size of the corpus (approx. 400 hours of melodies) is close to a minimum size when the statistical methods are worth of use.

The melody models were trained on the data. The results show that regardless of the style, the size of training data plays the crucial role for the model performance. However, the reason may be that we did not gather enough training data for the particular styles and only after joining all the data we got a corpus of a reasonable data size.

A history of 10 notes seems to provide the best trade-of between the n-gram order and performance of the model. According to [3] it might be enough to capture the hypothetical meaning of music.

It also turned out that Renaissance music is very different from all subsequent music styles, both of classical and popular music. It was the only style for which despite having relatively little training data the style specific model performed better than the model from all of the data. It is probably because the tonality which is common from Baroque times up today (major and minor scales) was not fully developed at the time of Renaissance. Furthermore Renaissance music is usually simpler in harmony, therefore the melody extraction works better on it. This might support the idea that the melody models use also implicit harmonic information hidden in the melodies.

3. Development of an Audio Melody Extraction System

In this section, development of an AME system is described. First a brief introduction to the current techniques is given. Basic ideas of our approach, its detailed description and evaluation follows.

The AME algorithms in the MIREX contest usually consist of two parts. The pitch detection and pitch tracking. Sometimes also separate voicing detection is used. The pitch detection algorithms often rely on the fact that the melody is usually sang by human voices [45]. In such methods spectra are searched for particular structures of harmonic frequencies in order to estimate the fundamental frequency. Other algorithms use series of very complicated filters combined with clever heuristics based on the rich experience with DSP. The erroneously detected pitches are then tracked by various dynamic programming algorithms based on some heuristic rules. Often the Hidden Markov Model (HMM) formalism is employed. The hidden states of such model represent the melody and the measured signal properties are the observation. Usually hand-written probabilities are used. Recently, some machine learning approaches also appeared [46] [47] as bigger datasets became available.

Our approach differs from all previously mentioned and relies on an exhaustive search of a hypotheses space inspired by the speech recognition and the statistical machine translation. Originally, we intended to use just a simple peak detection to estimate melody pitch candidates, later some other methods were utilized. Consecutive pitches of similar frequencies are grouped to tracks from which the hypotheses are generated. The resulting melody is the best-scoring sequence of hypotheses given the signal properties and the symbolic melody model. Except for the simple peak detection we use harmonic spectrum product, cepstrum-biased harmonic spectrum product and also a relatively complex algorithm of fundamentals frequencies detection. The last two mentioned method reach accuracy comparable with the algorithms used in the contest.

The approach can be intuitively interpreted as avoiding bringing prior knowledge while hand-crafting the probabilities to HMMs. We tried to keep the DSP as simple as possible and gain most of the intelligent behavior from the melody model.

All the DSP was implemented from scratch in the Scala programming language. Scala is a multi-paradigm programming language strongly preferring functional programming and is compiled to Java bytecode. We have chosen the language because it is very natural to think about DSP functionally, even though sometimes it is not very computationally efficient. Implementing everything from scratch was supposed to help us to have the full control of all parts of the algorithms and to avoid acquiring components without fully understanding them. Despite using Apache Commons library for linear algebra algorithms, the computations are still significantly slower than the same computations would be in Octave. A visualization tool was developed to have better view how the individual parts of the system work. Screenshots from this tool are used in the following sections to show how the presented algorithms work.

The algorithm could be possibly used on-line on a sufficiently fast machine because all computations can be done in linear time and depend only on the signal history. To keep the implementation simple, the system is not ready to be used on-line now. Re-implementing the algorithm in such way would also decrease memory requirements which are now linear with analyzed file size.

3.1 Pitch Detection

As a first step of the algorithm the pitch detection is performed. All the following methods use spectrogram computed using the Short Time Fourier Transform with window size of 4096 samples and window shift of 2048 samples. Each signal frame is zero-padded to twice of its length and is transformed by the Hanning window before the transformation. Algorithms are presented from the simplest and worst performing one to the most complex giving the best results. The last algorithm does not entirely fit to the idea of very simple DSP with most of intelligence incorporated in the hypotheses search, but it gives the best results.

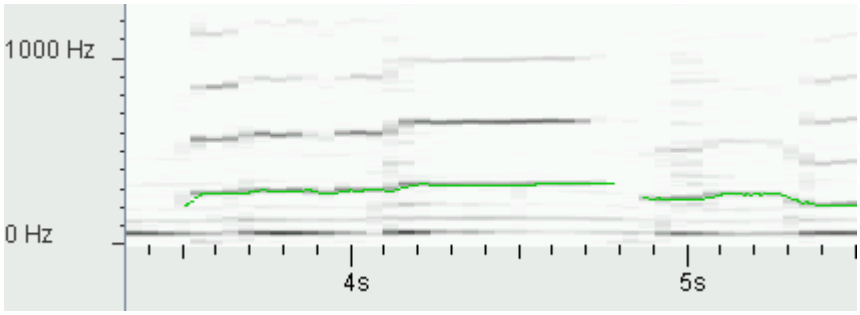


Figure 3.1: A spectrogram cut-out from `train01.wav`, the correct melody line is plotted in green.

For each algorithm a visualization of the algorithm result is provided. The same spectrogram cut-out of one file from the MIREX development set was used for each algorithm. The spectrogram cut-out with plotted melody frequencies is in Figure 3.1.

3.1.1 Simple Peak Detection

A very simple algorithm for peak detection was used. In each spectrum a local maximum f_{lm} of amplitude a_{lm} is recognized as a peak if it is a maximum in an interval $[f_1, f_2]$ where f_1 and f_2 are the closest frequencies to f_{lm} such that their amplitudes a_1 and a_2 satisfy inequality $a_i \leq a_{lm} - \delta$ where δ is the sensitivity of the algorithm. An example of how the algorithm works is plotted in Figure 3.2.

This algorithm performed very poorly for our task. In generated a lot of false candidates and it also often missed the correct pitch.

3.1.2 Harmonic Spectrum Product

The Harmonic Spectrum Product (HSP) is one of the oldest methods of pitch estimation, originally introduced in 1969 [48]. The method is based on the idea

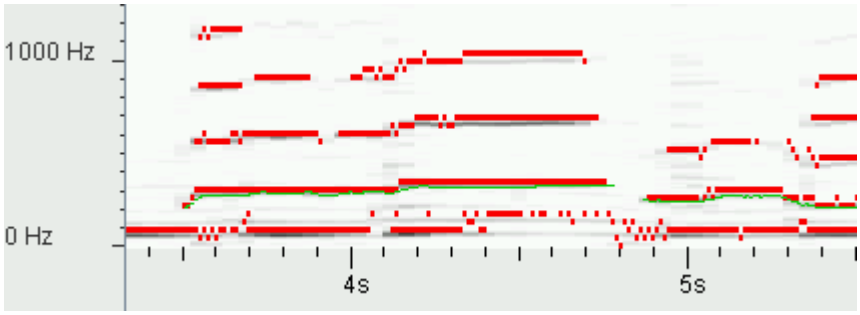


Figure 3.2: Peak detection (red lines) on a spectrogram cut-out from `train01.wav`, the correct melody line is plotted in green.

that the pitch is characterized not only by a signal peak, but also by the structure of its harmonics. When a spectrum is down-sampled such that each n-th sample is taken from the original spectrum, the peak of the n-th harmonic frequency should appear on the same index as its fundamental frequency in the original spectrum. By taking a sample-wise product of the down-sampled spectra we can observe a peak at the position of the fundamental pitch frequency.

For the k-th bin in a discrete spectrum we can write:

$$HSP(k) = \prod_{r=1}^{R} |X(kr)|^2, \quad 0 \leq k \leq N.$$

The frequency bin containing the pitch frequency is then estimated as:

$$\hat{k} = \arg\max_i HPS(i), \quad 0 \leq i < \frac{N}{R}.$$

The algorithm is very computationally efficient, though it does not perform particularly well in case of music. An example of how the algorithm result is plotted in Figure 3.3.

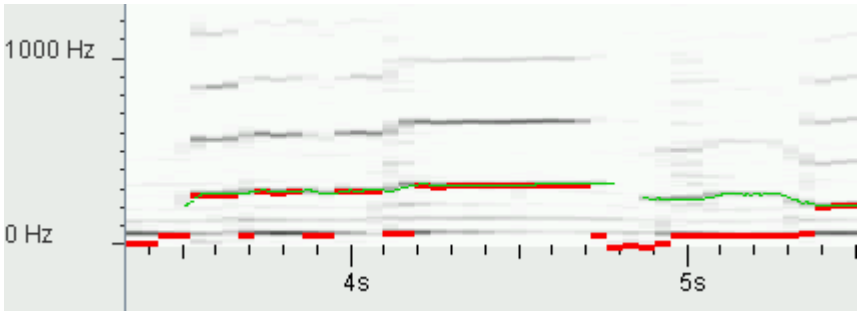


Figure 3.3: Harmonic Spectrum Product (red lines) on a spectrogram cut-out from `train01.wav`.

3.1.3 Cepstrum-Biased Harmonic Spectrum Product

The CBHSP algorithm[49] is a significant improvement of the HSP algorithm which combines the method with the cepstral analysis. Computing the fundamental frequency from the real cepstrum is a successful method in the speech

pitch estimation, but it performs poorly in case of music. The combination of cepstral analysis and HSP appeared to be very beneficial for both the speech and music pitch estimation.

In this particular case, we define the real cepstrum of the n-th signal frame as:

$$ceps(n) = |\text{IDFT}\left(\log|X(n)|\right)|,$$

where IDFT is the inverse Fourier transform.

Because the cepstrum is indexed in the time domain (the quefrencies can be interpreted as periods), we need to transform it to the frequency domain. For that purpose the value of the k-bin of the frequency indexed cepstrum is defined as:

$$FIC(k) = \max\left\{ceps(i)|0 \leq n < N, k = \left\lfloor\frac{N}{n}\right\rfloor\right\}$$

The CBHSP is then computed as a sample-wise product of the frequency indexed cepstrum and HSP. The frequency bin with the maximum CBHSP value is used as the melody fundamental frequency estimate. An example of the algorithm result is plotted in Figure 3.4.

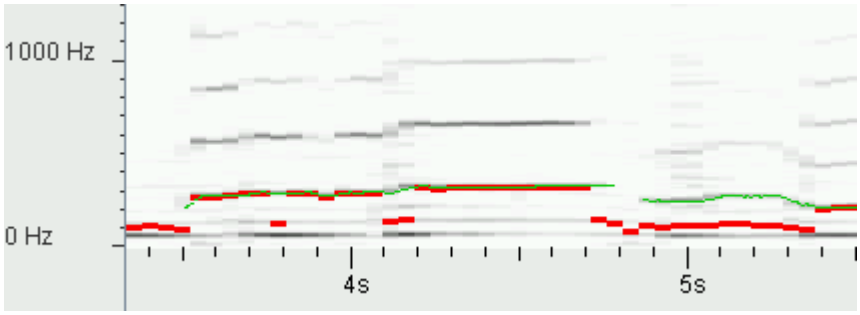


Figure 3.4: Cepstrum Biased Harmonic Spectrum Product (red lines) on a spectrogram cut-out from `train01.wav`.

3.1.4 Salience Function Maximization on Whitened Spectrogram

The most complex method we used for the pitch detection is a method for fundamental frequencies detection using an optimization of a salience function on a filtered spectrogram [50]. The salience function should capture how likely is a frequency to be a fundamental frequency in the spectrum given the frequencies at positions of expected harmonics.

For purposes of this algorithm we redefine the problem from finding a frequency f_f to finding a period τ with relation $f_f = f_s/\tau$, where f_s is the sampling frequency of the signal.

The fundamental period is then found by searching for the maximum value of the salience function

$$s(\tau) = \sum_{m=1}^{M} g(\tau, m) \cdot |Y(f_{\tau,m})|, \tag{3.1}$$

where $f_{\tau,m} = mf_s/\tau$ is the m-th harmonic frequency for the period τ, Y is the Fourier transform of the signal filtered by signal whitening (described in the following paragraph) and $g(\tau, m)$ is the weight of the m-th harmonic frequency in a spectrum with fundamental period τ.

The purpose of using filtered spectra Y is to suppress differences in spectra caused by different timbre of musical instruments and human voices. This approach is very similar to one that the author of [50] introduced in [51]. In that paper a perceptually motivated algorithm is proposed. First the equivalent rectangular bandwidth filter is applied to simulate the properties of hair cells in the inner ear. Afterwards the signal is flattened and the negative half-waves are eliminated in order to simulate the transmissive characteristics of the auditory nerve. Then a salience function similar to Equation 3.1 is applied to find the fundamental frequency, without explicitly knowing how the auditory nerve signal is processed

In order to get the whitened spectra, we first compute the sequence of 30 frequency centroids covering logarithmically the whole bandwidth of the spectrum with frequencies $c_b = 229 \cdot (10^{(b+1)/21.4} - 1)$ for $b = 1...30$. Then for each centroid c_b function triangular of H_b is defined as:

$$H_b(f) = \begin{cases} \frac{f - c_{b-1}}{c_b - c_{b-1}} & c_{b-1} < f \leq c_b \\ \frac{f - c_{b+1}}{c_b - c_{b+1}} & c_b < f < cb + 1 \\ 0 & \text{otherwise} \end{cases}$$

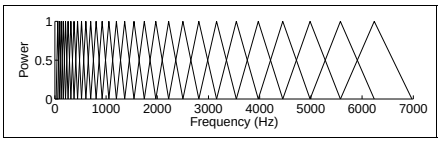


Figure 3.5: Plot of function H_b for all b, copied from [50]

Then function γ_b is defined as

$$\gamma_b = \left(\frac{1}{K} \sum_{k=0}^{K-1} H_b(f_k) |X(k)|^2 \right)$$

We compute the k-th frequency bin of the whitened spectrum Y as

$$Y(k) = \gamma(k) X(k),$$

where γk is computed by linear interpolation of values of $\gamma_{\lceil k \rceil}$ and $\gamma_{\lfloor k \rfloor}$.

The constants were chosen based on empirical evaluation which is described in more detail in [50].

Because we work with the discrete spectra approximated with the finite number of frequency bins, we need to use an approximation of the salience function. We do not know in which bin the corresponding harmonic frequency exactly is,

therefore we define a set of all possible candidates to be the m-th harmonic frequency belonging to fundamental frequency τ as

$$\kappa_{\tau,m} = \left\langle \text{round}\left(\frac{mK}{\tau + \frac{\Delta\tau}{2}}\right), \text{round}\left(\frac{mK}{\tau - \frac{\Delta\tau}{2}}\right)\right\rangle \quad \cap \quad \mathbb{N}.$$

Then the approximation of salience function can be expressed as

$$\hat{s}(\tau) = g(\tau, m) \max_{k \in \kappa_{\tau,m}} |Y(k)| \tag{3.2}$$

The g function which captures the relative amplitude of the m-th harmonic frequency in a whitened spectrum was machine learned and estimated as

$$g(\tau, m) = \frac{f_s/\tau + \alpha}{m f_s/\tau + \beta},$$

with parameters $\alpha = 27$ Hz, $\beta = 320$ Hz.

There also exist an efficient algorithm to find the maximum of the salience function based on splitting the period intervals. For that purpose the following salience function estimate for interval (τ_1, τ_2) is used

$$\sum_{m=1}^{M} \frac{f_s/\tau_1 + \alpha}{m f_s/\tau_2 + \beta} \cdot \max_{k \in \bigcup_{\tau',m} \kappa_{\tau',m}} |Y(k)|,$$

where τ' are values between τ_1 and τ_2 with a given step. The pseudocode of this algorithm is in Algorithm 3.1.

Algorithm 3.1 Searching for fundamental frequency using the bisection method

$Q \leftarrow \{(\tau_{\min}, \tau_{\max})\}$
$q_{\text{best}} \leftarrow (\tau_{\min}, \tau_{\max})$
while $\tau_{\text{up}} - \tau_{\text{low}} > \Delta\tau$, where $(\tau_{\text{low}}, \tau_{\text{up}}) = q_{\text{best}}$ **do**
  $Q \leftarrow Q \setminus \{q_{\text{best}}\}$
  $(\tau_{\text{low}}, \tau_{\text{up}}) \leftarrow q_{\text{best}}$
  $\tau_{\text{middle}} \leftarrow (\tau_{\text{low}} + \tau_{\text{up}})/2$
  $Q \leftarrow Q \cap \{(\tau_{\text{low}}, \tau_{\text{middle}}), (\tau_{\text{middle}}, \tau_{\text{up}})\}$
  $q_{\text{best}} \leftarrow \arg\max_{(\tau_1,\tau_2)\in Q} \hat{s}(\tau_1, \tau_2)$
end while
return $(\tau_{\text{low}} + \tau_{\text{up}})/2$, where $(\tau_{\text{low}}, \tau_{\text{up}}) = q_{\text{best}}$

In the algorithm the spectrum we gradually split into smaller intervals for which we estimate the maximum values of the salience function. We split the interval with the maximum value salience function estimate into two and recompute the salience function estimate for the new intervals. The algorithm ends when the maximum valued interval is smaller than a given threshold.

Due to its complexness, this method is the slowest one from all methods mentioned above. However, it performs best from the tested algorithms. For more detailed results see Section 3.3. An example of the algorithm result is plotted in Figure 3.6.

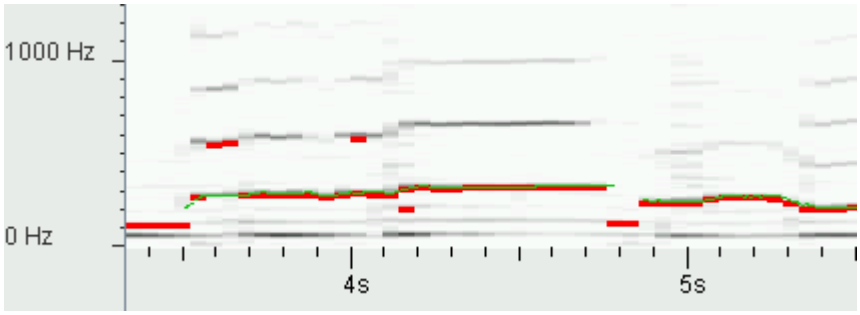


Figure 3.6: Salience function maximization results (red lines) on a spectrogram cut-out from `train01.wav`.

3.1.5 Combination of CBHSP and Salience Function

We tested a combination of the two most successful methods as well. Only fundamental frequencies which were detected both by CBHSP and the salience function maximization are taken into account. A tolerance of 0.4 semitone is used. This is supposed not to produce any output in the more confusing signal segments. The resulting gaps in the melody lines are supposed to be overcome in the tracking phase of the algorithm. An example of the algorithm result is plotted in Figure 3.7.

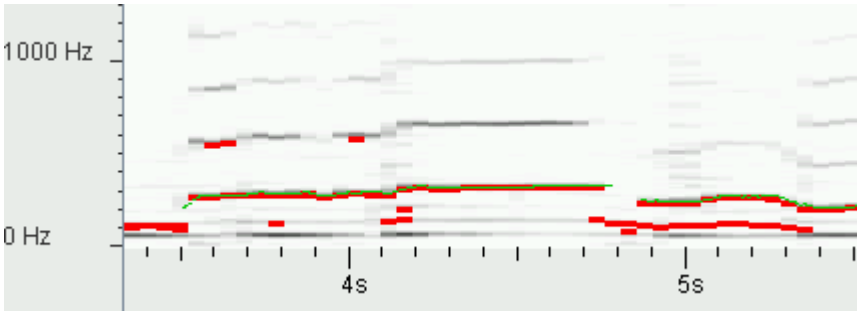


Figure 3.7: Salience function maximization and CBHSP combination results (red lines) on a spectrogram cut-out from `train01.wav`.

3.2 Searching the Extracted Tracks for the Target Melody

3.2.1 Partial Tracking

We cluster the pitch frequency candidates using the partial tracking algorithm [52]. It is a simple algorithm to find sequences of consecutive spectral peaks and connect them into tracks. The spectrogram is processed iteratively frame by frame from the lower frequencies to the higher ones.

Now suppose that all frames up to the k-th one been processed. A frequency peak p representing frequency f_p is added to a track t if it satisfies inequality

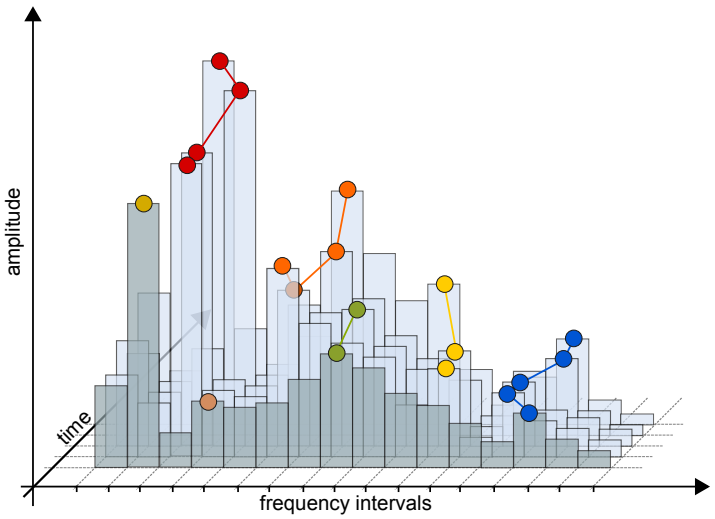


Figure 3.8: An illustration of the partial tracking algorithm. Detected spectral peaks with similar frequencies are being connected into tracks.

$$|f_p - f_{t_p}| < \Delta_f$$

where f_{t_p} is frequency of the previous peak included in the track t. We terminated tracks that have not been updated for longer period than e frames. We remove the terminated tracks which are shorter than l frames and those with proportion of the non-empty frames lower than d. Frequency peaks that have not been linked to any of the previous tracks give raise to new ones. We use the multiplies of 12 of the difference of the binary logarithms which represents by how many semitones the pitches differ.

There are various both purely numerical and statistical methods to improve performance of this algorithm. They could be used for attempts for further improving the task.

The algorithm parameters we chose for the particular pitch detection methods are tabulated in Table 3.1.

	Δ_t	e	l	d
peak detection	0.8	12	0.9	7
HPS	0.6	11	0.9	6
CBHPS	0.7	3	0.7	1
salience function	0.8	3	0.5	2
combined	0.6	12	0.2	1

Table 3.1: Results of parameter optimization of the partial tracking

3.2.2 Track Interpretation Generation

When the partial tracks are identified in the spectrogram, their possible interpretations are generated. A set of rules is used to generate scored possibilities of the

indication	meaning	value
op	octave mismatch penalty	0.5
dt	duration tolerance	0.01
Δd	duration step	0.05 s

Table 3.2: Parameters used for track interpretation generation

tracks interpretation. First, the pitch of the note is estimated. We use the following formula to compute the MIDI notation pitch number from the frequency mean:

$$pitch(f) = 69 + 12 \log_2 \left(\frac{f}{440 \text{ Hz}} \right).$$

Score $s_1 = |1 - \frac{pitch(f)}{\lfloor pitch(f) \rfloor}|$ is assigned to the pitch of $\lfloor pitch(f) \rfloor$ and analogically score s_2 is assigned to pitch $\lceil pitch(f) \rceil$. To include also octave mismatches, an octave lower and octave higher pitches are assigned a score of $op \cdot s_1$ and $op \cdot s_2$ respectively. The pitch scores are normalized to sum up to one.

For the note durations possible start and end times are generated within the range of d times track duration around the track start and end time. Time intervals around are split to discrete steps of size Δd. To all possible combination of start and end time a score is assigned:

$$s(start, end) = \left| 1 - \frac{end - start}{d} \right|,$$

where d is the track duration. Only the duration of the note, not the exact position is included in the score. We expect the melody model to adjust its exact temporal position with respect to the surrounding rhythmic pattern. The duration scores are normalized to sum up to one.

Finally, a list of scored note hypotheses is created from the track by computing all possible combinations of pitches and duration and assigning them a score equal to multiply of the pitch and the duration score. Parameters used in the algorithm are tabulated in Table 3.2.

3.2.3 Searching the Hypotheses Space

For searching for the sequence of notes representing the melody we use an algorithm which can be briefly described as pruned breadth-first search.

The algorithm is described in detail in pseudocode Algorithm 3.2. An object-like notation is used in the pseudocode. A track has its start and end, a processed track also contains a list of hypotheses of note sequences ending with a note generated from that particular track. An unprocessed track has method getPossibleNotes which generates a list of scored notes (as was described in the previous section). Scores are supposed to be expressed by the logarithms of probabilities, or more precisely score which sum up to one. Parameters used in the algorithm are tabulated in Table 3.3.

During the algorithm run, tracks are kept in two sets – a set of unprocessed tracks and a set with already processed tracks containing n-best lists of hypotheses of note sequences leading to notes generated from the tracks. When a track is

Algorithm 3.2 Algorithm searching for the most likely tone sequence

k ... number of previous track taken in acount
m ... number of the best scoring hypotheses kept in processed tracks
l ... number of hypotheses at the begining and at the end of the analysed sound that can be used to start or end resulting melody

$T \leftarrow$ tracks sorted by start time
$P \leftarrow \varnothing$ ▷ set of processed tracks
while $T \neq \varnothing$ **do**
  $t \leftarrow t' \in T$ with minimum start time
  $pred \leftarrow P$.sortBy(p.end - t.start, $p \in P$).take(k)
  $newHypotheses \leftarrow \varnothing$
  for all $p \in \bigcup_{p \in pred} p$.hypotheses **do**
    for all $n \in t$.getPossibleNotes **do**
      $newHypotheses \leftarrow$
        $newHypotheses \bigcup \{(h..\text{notesADD}(n.\text{note}), h.\text{score} + n.\text{score})\}$
    end for
  end for
  $normedNewHyp \leftarrow$norm($newHypothese$ with score $s'(h) = \frac{h.\text{score}}{|h.\text{notes}|^{\alpha}}$)
  $recoredHypotheses \leftarrow$ rescoreWithMelodyModel($normedNewHyp$)
  $P \leftarrow P \bigcup (t, rescoredHypothese.\text{take}(m))$
  $T \leftarrow T \setminus \{t\}$
end while
return rescoreWithMelodyModel(
    P.sortBy(p.track.end, $p \in P$).take(l)).first

indication	meaning	value
k	number of previous track to be connected with the processed ones	3
m	number of best scoring hypotheses kept in processed tracks	20
l	number of edge tracks considered to be the starting or ending ones	5
α	hypothesis density bonus	3

Table 3.3: Parameters used for track interpretation generation

being processed all notes generated from the track are used to extend all the hypotheses from k previous tracks. This step gives raise to new hypotheses whose scores are sums of the hypotheses scores and the added notes scores (note that logarithms of scores are used). This is illustrated in Figure 3.9.

If only this value (sum of logarithms of the note scores) would be used as the hypotheses score it would lead to preferring hypotheses with high proportion of silence. Therefore we divide the score by the number of notes raised to the power of α, where α is a parameter expressing the preference for denser sequences.

This hypotheses scores are then normalized to sum up to one and rescored using the melody model. Only m best hypotheses after the model rescoring are stored together with the track in the set of processed tracks.

Finally, hypotheses leading to the l last tracks are evaluated by the melody model and the one having the best score is chosen to be the final solution.

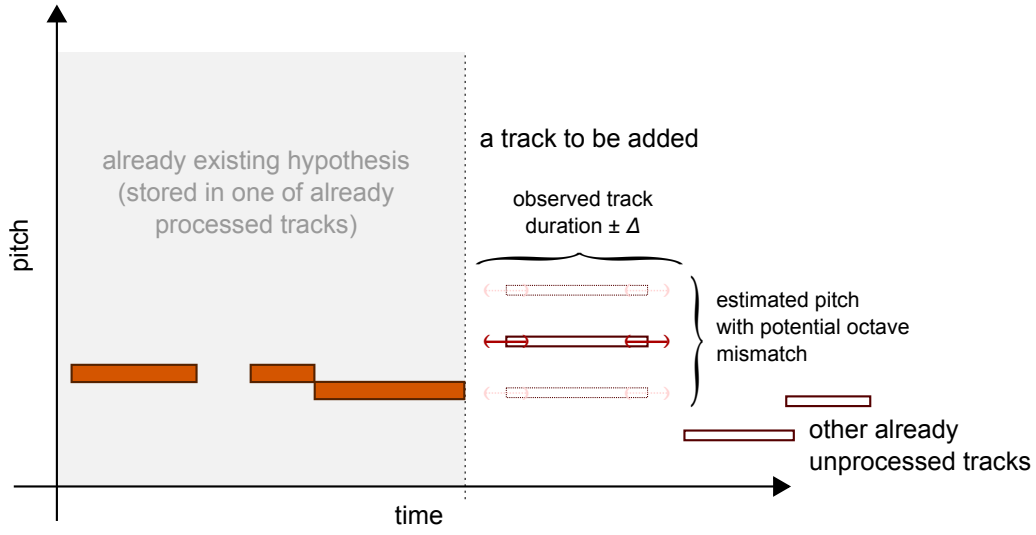


Figure 3.9: An illustration of one step of the hypotheses generation

3.3 System Evaluation

All the pitch detection algorithms except the simple peak detection perform almost perfectly in the very simple cases as singing in silence or singing with a simple guitar accompaniment. Therefore it was necessary to test the algorithm performance on a more challenging test set.

We have chosen the MIREX05 development data. It is a set of 13 excerpts of popular music with melody sung by humans which claims to cover all possible problems which can appear in the AME task. It is also supposed to be very similar to the MIREX05 test set used for evaluation of algorithm participating in the MIREX. So we can roughly compare performance of our algorithm with the others. The test set itself is not publicly available.

The test data are very fine graded both in the resolution of time and the detected frequency. Therefore the tendency of our algorithm to produce an output more similar to symbolic notation than to this signal based representation may be a source of some errors.

The algorithm was tested in different setups – with different pitch detection algorithms and with different melody models.

Detailed results are tabulated in Table 3.4, even more detailed results with the score for each of the test files are tabulated in Attachment A.3. For each experiment the results contain the voicing accuracy (proportion of signal frames where it was correctly estimated if it contains melody or not) on the first line, raw pitch accuracy (proportion of melody signal frames where the pitch was estimated not further than 0.25 tone from the correct value) on the second line, raw chroma accuracy (proportion of melody signal frames where the pitch was estimated not further than 0.25 from the correct value, disregarding the octave mismatch) on the third line and overall accuracy (proportion of correctly marked silence frames or correctly estimated pitch in the whole music excerpt) on the fourth line.

The results show that the most suitable method of pitch detection method is the salience function maximization. It also turned out that suitable selection of the track search algorithm parameters leads to better results than using the melody model based on the symbolic data. Anyway, the model turned to be at least partially useful in case when all the hypotheses were assigned uniformly distributed score. The model based on unclustered data performed slightly better than the model assigning at least some probability all possible tokens. In the best setup we reached the voicing accuracy of 56 % and raw pitch accuracy of 23 %, which is very low in a comparison with usual 90 % voicing accuracy and 70 % raw pitch accuracy of the state-of-the-art algorithms. Reasons for this results are discussed in detail in the next section.

The computation time is relatively high. The DSP implemented in Scala is not as fast in Matlab or Octave which are commonly used for DSP experiments. The slowest part of the algorithm is the melody model rescoring even though the SRILM toolkit is used in the server-client configuration. The model is loaded in the the server process which can be quickly queried for the results.

We tried to apply further post-processing techniques to improve the system performance, always without any success. The first worth of mentioning is an attempt to align the generated notes to note onsets detected by a note onset detection implemented according to [53]. The second originally seemingly promising idea was filling big pauses by notes generated by the melody model. Both of these experiments the computation time significantly and they also did not raise the performance of the algorithm at all.

3.4 Discussion

At the beginning of the AME development we started with very simplistic and also very naive approaches how to implement an AME system. Almost all of our original beliefs turned to be false.

Even though it was formulated as a little complain in the previous section that the only available standard data does not really matches the problem as it was approached in this thesis, it seems very reasonable to formulate the AME problem such that it should give output in the form of the provided data. When we look at the data more carefully we can identify some moments when the melodic line differs by more than a half tone from a steady line, although listeners would interpret such moments just as tiny ornaments or interpretative imperfectness

melody model	without model	with rat0 model	with rat0-clust model	only rat0 model	only rat0-clust model
peak detection	0.378	0.392	0.395	0.392	0.394
	0.000	0.006	0.006	0.006	0.012
	0.011	0.018	0.012	0.018	0.022
	0.220	0.221	0.221	0.221	0.225
HSP	0.244	0.244	0.224	0.224	0.224
	0.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000
	0.224	0.224	0.224	0.224	0.224
CBHSP	0.558	0.484	0.486	0.484	0.486
	0.167	0.067	0.060	0.067	0.061
	0.196	0.114	0.097	0.114	0.101
	0.273	0.182	0.180	0.182	0.180
salience function	0.568	0.601	0.606	0.601	0.606
	0.231	0.104	0.056	0.104	0.063
	0.265	0.151	0.122	0.151	0.134
	0.247	0.153	0.117	0.153	0.123
combined	0.391	0.441	0.440	0.441	0.441
	0.143	0.082	0.037	0.082	0.047
	0.148	0.103	0.074	0.103	0.088
	0.284	0.229	0.198	0.229	0.203

Table 3.4: Results of AME algorithms on the MIREX05 dataset.

in a longer steady note. It would be a longer steady note also in the symbolic transcription and this is what the melody model expects as well. It is mostly this phenomenon together with utilizing the partial tracking algorithm which plays the crucial role for the low system accuracy on complicated data. Instead of helping to overcome the problems originating in the erroneous pitch detection as the tracking algorithms in AME system usually do, it tends to often separate pitches belonging together and to produce a lot of short chromatic tracks. The track searching algorithm with the melody model then expects something very similar to the actually symbolic notation and fails to recognize the best sequence of tracks which if transcribed to notes would be very unlikely.

The real goal of the AME task is to identify which frequencies are part of the melody perceived by humans no matter what abstract representation will the listeners assign to these frequencies. How a sequence of such frequencies will be perceived on a more abstract level is a different task. This difference between the *physical melody* and the *abstract melody* has not been taken in account during the algorithm development. On the other hand, all the literature dealing with

the melody from the DSP perspective does not distinguish this two senses of melody at all. Modeling the abstract melody given the physical melody belongs more to the field of cognitive science. It could be approached as a machine learning problem while having suitable training data. Nevertheless, dealing with this problem sufficiently, including the previous work research, would give enough material to produce another thesis.

Having made this experience we would suggest to approach this problem with a different algorithm, a much more similar to algorithm that succeeded in the MIREX contest like [54]. The HMMs are used to estimate the most probable melody. Melody is modeled as a sequence of hidden states and the emitted observations are some statistics computed from the signal. Given the observation the hidden states can be usually the observed pitch itself, octave mismatch, silence or a value approximation based on the surrounding hidden states.

The state transition probabilities usually use hand-written rules. Instead we could use a model trained on the development data. A suitable tool to estimate the emission probabilities could be a Bayesian network with the signal properties as terminal nodes. In fact, it would be a purely machine learning algorithm not employing any NLP methods.

If a model transcribing the physical melody to the abstract would be available for the proposed AME system, the current melody model could be used as a back-off model the transition probabilities. However, usefulness of integrating such model in the system is doubtful because of a danger of bringing more errors by combining not perfectly working components.

3.5 Summary

In this section a development of an AME system was described. Various methods of pitch detection were tested, with the CBHSP and perceptually motivated salience function maximization on a filtered signal giving the best results. In the next phase of the algorithm similarly pitched frames were grouped to tracks. The resulting sequence of notes was received by a pruned breadth-first search of the possible note interpretation of the tracks. The melody model was used to score the hypotheses for pruning in the search. The results showed that the scoring of the hypotheses using fixed parameters leads to better results than using a melody model. The salience function maximization turned to be the best pitch detection method. The reached voicing accuracy of 56 % and raw pitch accuracy of 23 % remains below the state-of-the-art algorithm performance.

The system scored worse than most of the recently published algorithms, mostly because of a conceptual misunderstanding of a difference between symbolically represented melody in an abstract way and melody in a more physical sense as a sequence of consecutive frequency values. On the other hand, the experiments that have been made contributed to a better definition of the goal of the AME in the DSP by distinguishing how it differs from what is commonly referred to be a melody.

Conclusion

The goal of the thesis was to summarize similarities between music and natural languages with respect to their computational processing, make experiments with some of the NLP inspired methods of music processing and develop a practical application showing applicability of such methods.

A lot of theoretical research has been done since 1950s which gives a satisfactory foundation for computational applications. The first experiments appeared in 1950s and focused on machine composition or composer and style detection. All work in this field closely follow the paradigm development in computational linguistics and another artificial-intelligence-like fields of study with a radical shift from the systems based on hand-written rules to purely statistical systems in 1990s. As in other fields which are part of the humanities, alternating periods of optimism and skepticism towards quantitative and computational methods appeared and influenced the amount of work published on such topics in different periods of time. In more theoretical works many hypotheses about the relation between language and music have been raised and computational methods have potential to bring supporting or rejecting arguments for them.

A corpus of MIDI files of various styles of total length of 717 hours was collected. State-of-the-art methods of melody extraction from MIDI files were discussed and an algorithm combining various known techniques was used to create a corpus of extracted melodies. This corpus consists of 408 hours of melodies.

From this data a melody model using the language modeling techniques was created. Various methods of melody representation were tested to increase the robustness of the model. The best way of melody representation for this purpose was to use only the differences between adjacent notes represented as pairs of the pitch interval and ratio of their durations. The duration ratio is rounded to integers in case it is bigger than one and as a fraction with numerator of one and denominator rounded to an integer if it is smaller than one. Differences occurring only rarely in the corpus were grouped to several clusters, to eliminate the problem of out-of-vocabulary tokens.

Despite the melody extraction being far from perfect, the melody models seem to provide very promising results while tested on melodies extracted from MIDI files using the same melody extraction algorithm. Results of measuring the perplexity of the model on test sets of various styles (styles of classical music from Renaissance to 20^{th} century, jazz and popular music) show that more important than the style correspondence is the amount of training data. The only exception from this observation was the Renaissance music which is fundamentally different from all the later styles.

In the last part of the thesis development of an alternative AME system is described. The audio signal is first transformed to the frequency domain, pitch candidates are detected in each spectrum, similar consecutive pitch candidates are gathered to tracks using the partial tracking algorithm and the tracks are searched for the best sequence using the melody model.

The original idea that a simple peak detection in frequency domain in combination with the exhaustive search using the melody model will be enough turned out to be very naive. Much more complex methods for pitch detection must have

been employed to achieve at least acceptable accuracy. The performance of the system measured on the standardized MIREX data remains below the state-of-the-art techniques. Both the partial tracking algorithm and the melody model expect the melody to be much more similar to the symbolic representation than it actually was. This helped us to reveal a conceptual difference between the melody understood from the DSP perspective as a sequence consecutive pitch frequencies and melody perceived on an abstract level.

The whole AME system is implemented in Scala programming language. General signal processing classes can be used as a base of Scala DSP library.

Unfortunately, most of the methods we originally planned to experiment with, most of them summarized in Section 2.3, remained unexplored. The thesis probably should have focused on a much narrower topic which could have been studied to much bigger details and lead to more interesting results.

Still, the biggest contribution of the thesis is bringing a coherent overview of the recent history of music processing and music and language relationship from the perspective of computational linguistics. A melody corpus for the melody has been created and is ready for further experiments. Finally, the experiments with the AME system, despite not being successful in the AME task itself, helped us define the difference between physical and abstract melody.

Bibliography

[1] Limb, Charles J. – Braun, Allen R. Neural Substrates of Spontaneous Musical Performance: An fMRI Study of Jazz Improvisation. [online] In: *PLoS ONE, http://www.plosone.org/* Public Library of Science, 2008. [cit. 22.6. 2011] <http://www.plosone.org/article/info:doi%2F10.1371%2Fjournal.pone.0001679>

[2] Dobrian, Christopher. *Music and Language* [online]. Irvin (California, USA). Universtiy of California, 1992. [cit. 19.3. 2011] <http://music.arts.uci.edu/dobrian/>

[3] Meyer, Leonard B. Meaning in Music and Information Theory. In *The Journal of Aesthetics and Art Criticism, Vol. 15, No. 4 (Jun., 1957).* Hoboken (New Jersey, USA). Blackwell Publishing, 1957. pp. 412-424.

[4] Manzara, Leonard C. – Witten, Ian H. – Mark, James. On the Entropy of Music: An Experiment with Bach Chorale Melodies. In: *Leonardo Music Journal, Vol. 2, No. 1 (1992)* Cambridge (Massachusetts, USA). The MIT Press, 1992. pp. 81-88.

[5] Lerdahl, Fred – Jackendoff, Ray. *A Generative Theory of Tonal Music* Cambridge (Massachusetts, USA). The MIT Press, 1983.

[6] Wołkowicz, Jacek – Kulka, Zbigniew – Kešelj, Vlado. *n*-gram-based Approach to Composer Recognition. In: *Archives of Acoustics, Vol. 33 (2008).* Warszawa (Poland). Polish Academy of Sciences, 2008.

[7] Bod, Rens. Using Natural Language Processing Techniques For Musical Parsing. In: *In: Proceedings Association for Computers and the Humanities and the Association for Literary and Linguistic Computing 2001.* Springer, 2001.

[8] Bernstein, Leonard. *The unanswered question: six talks at Harvard.* Cambridge (Massachusetts, USA). Harvard University Press, 1976.

[9] Dubnov, Shlomo – Gerard, Assayag – Olivier, Lartillot – Gil, Bejerano. Using machine-learning methods for musical style modeling. In: *IEEE Computer Society, Vol. 36, No. 10 (Oct. 2003).* IEEE Computer Society Press, 2003. pp. 73-80.

[10] Meehan, James R. An Artificial Intelligence Approach to Tonal Music Theory. In *Computer Music Journal, Vol. 4, No. 2 (Summer 1980), Artificial Intelligence and Music Part 1.* Cambridge (Massachusetts, USA). The MIT Press, 1980. pp. 60-65.

[11] Wang, Jun – Chen, Xiaoou – Hu, Yajie – Feng, Tao. Predicting High-level Music Semantics using Social Tags via Ontology-based Reasoning. In: *Proceedings of 11th International Society for Music Information Retrieval Conference.* International Society for Music Information Retrieval, 2010.

[12] Shannon, Claude E. Prediction and entropy of printed English. In: *Bell system technical journal, Vol. 30, No. 1 (1951). pp. 50-64.* Bell Labs, 1951.

[13] Dobrian, Christopher. *Music and Artificial Intelligence* [online]. Irvin (California, USA). Universtiy of California, 1993. [cit. 19.3. 2011] <http://music.arts.uci.edu/dobrian/>

[14] Witten, Ian H. – Manzara, Leonard C. – Conklin, Darrell. Comparing Human and Computational Model of Music Prediction. In: *Computer Music Journal, Vol. 18, No. 1 (Spring, 1994).* Cambridge (Massachusetts, USA). The MIT Press, 1994. pp. 70-80.

[15] Mauch, Matthias – Müllensiefen, Daniel – Dixon, Simon – Wiggins, Geraint. Can Statistical Language Models be used for the Analysis of Harmonic Progressions? In: *International Conference on Music Perception and Cognition Proceedings 2008.* Hokkaido (Japan). Hokkaido Univeristy, 2008.

[16] Zanette, Damian H. Zipf's law and the creation of musical context. In: *Musicae Scientiae, Vol. 10, No. 1.* European Society for the Cognitive Sciences of Music, 2006.

[17] Ringer, Alexander. Melody. In: *G*rove Music Online ed. L. Macy [online] Oxford University Press, 2013. [cit. 5.3.2013] <http://www.grovemusic.com>

[18] Eco, Umberto. Skeptikové a těšitelé. (Czech translation) Praha (Czech Republic). Argo, 2006.

[19] Pearce, Marcus T. *Early Applications of Information Theory to Music.* [online] University of London, 2007. [cit. 17.7. 2011] <http://www.doc.gold.ac.uk/∼mas01mtp/notes/music-information-theory.pdf>

[20] Mauer, John. *A Brief History of Algorithmic Composition.* [online] Stanford University, 1999. <http://ccrma.stanford.edu/∼blackrse/algorithm.html>

[21] Cont, Arshia. Artifficial Intelligence and Music: A critical survey and proposal. A result of research in the first year of Computer Music PhD program. San Diego (California, USA). University of California, 2005.

[22] Schaffrath, Helmut – Huron, David. *The Essen folksong collection in kern format.* [computer database] Menlo Park (California, USA). Center for Computer Assisted Research in the Humanities, 1995.

[23] Suyto, Iman S. H. – Uitdenboerg, Alexandra L. Simple Efficient n-gram Indexing for Effective Melody Retrieval. In: *6th International Conference on Music Information Retrieval.* University of London, 2005.

[24] Wołkowicz, Jacek – Kešelj, Vlado. Text Information Retrieval Approach to Music Information Retrieval. In: *The Music Information Retrieval Evaluation eXchange Submitions* [online] International Society for Music Information Retrieval, 2011. [cit. 10.3. 2013] <http://www.music-ir.org/mirex/abstracts/2011/WK1.pdf>

[25] Doraisamy, Shyamala – Rüger, Stefan. A Polyphonic Music Retrieval System Using n-grams. In: *Proceedings of the 5th International Conference on Music Information Retrieval.* International Society for Music Information Retrieval, 2004.

[26] Urbano, Julian, Lloréns, Juan et al. MIREX 2012 Symbolic Melodic Similarity: Hybrid Sequence Alignment with Geometric Representations. In: *The Music Information Retrieval Evaluation eXchange Submitions* [online] International Society for Music Information Retrieval, 2012. [cit. 10.3. 2013] <http://www.music-ir.org/mirex/abstracts/2012/ULMS2.pdf>

[27] Mozer, Michael C. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. In: *Connection Science, Vol. 6, No. 2-3 (1994), pp. 247-280.* Taylor & Francis, 1994.

[28] Hori, Toyokazu – Shinichiro, Wada – Howzan, Tai – Kung, S. Y. Automatic music score recognition/play system based on decision based neural network. In: *3rd IEEE Workshop on Multimedia Signal Processing, pp. 183-184.* IEEE Computer Society, 2005.

[29] Zicarelli, David. M and Jam Factory. In: *Computer Music Journal, Vol. 11, No. 4 (Winter, 1987), pp. 13-29.* Cambridge (Massachusetts, USA). The MIT Press, 1987.

[30] Pachet, François. The Continuator: Musical Interaction With Style. In: *International Computer music Conference, Gotheborg (Sweden), (ICMA 2002).* Ann Arbor (Michigan, USA). MPublishing (University of Michigan), 2002.

[31] Rizo, David – León, PJ Ponce de – Pérez-Sancho, C. A Pattern Recognition Approach for Melody Track Selection in Midi Files. In: *Proceedings of the 7th International Society for Music Information Retrieval Conference.* International Society for Music Information Retrieval, 2006.

[32] Uitdenbgerd, Alexandra – Zobel, Justin. Melodic matching techniques for large music databases. In: *Proceedings of the seventh ACM international conference on Multimedia (Part 1) - MULTIMEDIA '99, pp. 57-66.* ACM, 2012.

[33] Ozcan, Giyasettin – Isikhan, Cihan – Alpkocak, Adil. Melody Extraction on MIDI Music Files. In: *Seventh IEEE International Symposium on Multimedia.* IEEE Computer Society, 2005.

[34] Robertson, Hannah. Slides for Music Technology Seminar course at McGill University, Montreal, Canada. [online]. McGill University, 2012. [cit. 11.3. 2013] <http://www.music.mcgill.ca/~hannah/MUMT621/robertson_mumt621_melody.html>

[35] Velusamy, Sudha – Thoshkahna, Balaji – Ramakrishnan, K. A Novel Melody Line Identification Algorithm for Polyphonic MIDI Music. In: *Lecture Notes in Computer Science – Advances in Multimedia Modeling, pp. 248-257.* Springer, 2006.

[36] Kosugi, Naoko – Nishihara, Yuichi – Kon'ya, Seiichi et al. Music retrieval by humming-using similarity retrieval over high dimensional feature vector space. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, 1999.* IEEE Computer Society, 1999.

[37] Wołkowicz, Jacek – Brooks, Stephen – Kešelj, Vlado. Midivis: Visualizing music structure via similarity matrices. In: *Proceedings of the International Computer Music Conference (ICMC), pp. 53–56.* Montreal, (Quebec, Canada). International Computer Music Association, 2009.

[38] Manning, Christopher D. – Schütze, Hinrich. *Foundations of Statistical Natural Language Processing.* Cambridge (Massachusetts, USA). The MIT Press, 1999.

[39] Arisoy, Ebru – Sainath, Tara N. – Kingsbury, Brian – Ramabhadran Bhuvana . Deep Neural Network Language Models. In: *WLM '12 Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pp. 20-28. Association for Computational Linguistics, 2012.

[40] David, Temperley. A Probabilistic Model of Melody Perception. In: *Cognitive Science, Vol. 32, No. 2 (March 2008), pp. 418–444.* Cognitive Science Society, 2008.

[41] Bikel, Daniel M. – Schwartz, Richard – Weischedel, Ralph M. An algorithm that learns what's in a name. In: *Machine learning Vol. 34, No. 1 (1999), pp. 211-231.* Springer, 1999.

[42] Stolcke, Andreas. SRILM-an extensible language modeling toolkit. In: *Proceedings of the international conference on spoken language processing. Vol. 2. 2002.* International Speech Communication Association, 2002.

[43] Mochihashi, Daichi – Yamada, Takeshi – Ueda, Naonori. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP. Vol. 1.* Association for Computational Linguistics, 2009.

[44] Rissanen, Jorma. *Stochastic complexity in statistical inquiry theory.* Singapore (Singapore). World Scientific Publishing Co., Inc., 1989.

[45] Chien, Yu-Ren – Wang, Hsin-Min – Jeng, Shyh-Kang An acoustic-phonetic approach to vocal melody extraction. In: *Proceedings of the International Symposium on Music Information Retrieval 2011.* International Society for Music Information Retrieval, 2011.

[46] Marolt, Matija. Gaussian mixture models for extraction of melodic lines from audio recordings. In: *Proceedings of the 5th International Society for Music Information Retrieval Conference, pp. 80-83.* International Society for Music Information Retrieval, 2004.

[47] Ellis, Daniel PW – Poliner, Graham E. Classification-based melody transcription. In: *Machine Learning, Vol. 65, no. 2-3 (2006), pp. 439-456.* Springer, 2006.

[48] Noll, A. Michael. Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate. In: *Proceedings of the Symposium on Computer processing communications. Vol. 779, 1967.* Polytechnic Institute of Brooklyn, 1969.

[49] Master, Aaron S. *Speech spectrum modelling from multiple sources.* Doctoral Dissertation, University of Cambridge, 2000.

[50] Klapuri, Anssi. Multiple fundamental frequency estimation by summing harmonic amplitudes. In: In: *Proceedings of the 7th International Conference on Music Information Retrieval.* International Society for Music Information Retrieval, 2006.

[51] Klapuri, Anssi. A perceptually motivated multiple-f0 estimation method. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.* IEEE Computer Society, 2005.

[52] McAulay, Robert – Quatieri, Thomas. Speech analysis/synthesis based on a sinusoidal representation. In: *IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 34, No. 4 (1986)* IEEE Computer Society, 1986. pp. 744-754.

[53] Alonso, Miguel – Gaël, Richard – Bertrand, David. Extracting note onsets from musical recordings. In: *IEEE International Conference on Multimedia and Expo, 2005.* IEEE Computer Society, 2005.

[54] Tzu-Chun, Yeh – Jyh-Shing, Roger – Jang, I-Bin Liao. Methods for Audio Melody Extraction in MIREX 2012. *The Music Information Retrieval Evaluation eXchange Submitions* [online] International Society for Music Information Retrieval, 2012. [cit. 20.3. 2013]

List of Tables

List of Figures

Acronyms

AME Audio Melody Extraction.

CBHSP Cepstrum Biased Harmonic Spectrum Product.

DSP Digital Signal Processing.

HMM Hidden Markov Model.

HSP Harmonic Spectrum Product.

MIR Music Information Retrieval.

MIREX Music Information Retrieval EXchange.

NLP Natural Language Processing.

A. Attachments

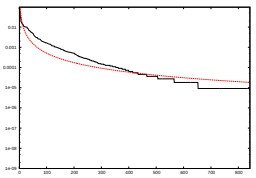
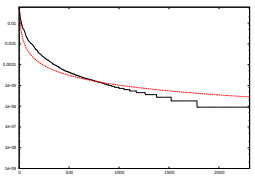
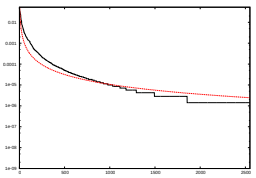
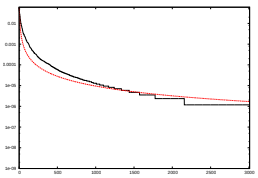
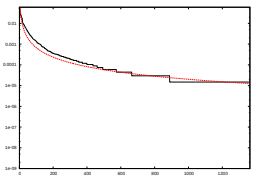
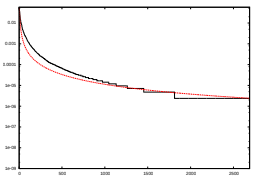
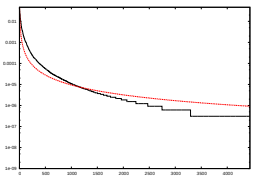
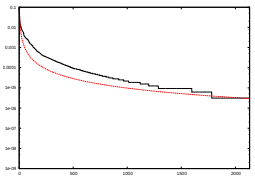
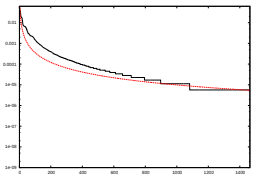
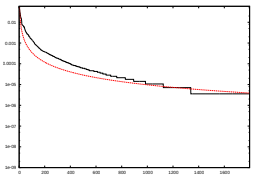
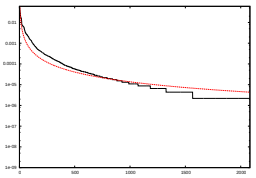
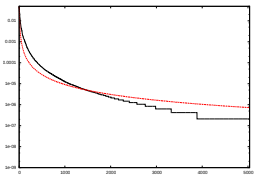
A.1 Melody and Zipf's-Mandelbrot's Law

In the following table the distributions of melody units sorted according to their frequencies are plotted (black). In each graph the best-fitting Zipfian curve is plotted (red). Parameters of the curve

$$f = P(r + \rho)^{-B}$$

are mentioned aside.

On the horizontal axis the ranks of the of frequencies are plotted, on the vertical axis, the frequencies in the logaritmic scale.

renaissance $P = 0.70$ $\rho = 0.07$ $B = 1.57$		baroque $P = 0.53$ $\rho = 0.23$ $B = 1.57$	
classicism $P = 0.55$ $\rho = 0.23$ $B = 1.57$		romanticism $P = 0.48$ $\rho = 0.23$ $B = 1.57$	
impressionism $P = 0.77$ $\rho = 0.04$ $B = 1.53$		20thcentury $P = 0.56$ $\rho = 0.21$ $B = 1.57$	
classical $P = 0.48$ $\rho = 0.23$ $B = 1.57$		jazz $P = 0.48$ $\rho = 0.23$ $B = 1.56$	
pop $P = 0.48$ $\rho = 0.23$ $B = 1.56$		rock $P = 0.47$ $\rho = 0.24$ $B = 1.56$	
popular $P = 0.70$ $\rho = 0.07$ $B = 1.57$		all $P = 0.47$ $\rho = 0.24$ $B = 1.57$	

A.2 Perplexity of Langugage Models on Test Data

The following tables contain detailed results of melody modeling experiments. The following abbreviations are used for methods of the notes changes representation from Section 2.2.2:

- *rat0* – In this notation both pitch intervals and duration ratios are expressed exactly as they were computed from the melodies.
- *rat0clust* – In this notation intervals bigger than 16 and smaller than -16 semitones are clustered to few groups. Duration ratios bigger than 8 and smaller than 1/8 are clustered as well.
- *rat0mod* – The intervals are expressed as the interval size modulo 12. The duration ratios are expressed in the same way as in the previous method.

From each data set (see data size overview in Table 2.2) 90 % randomly selected snippets was used as training data, the rest 10 % as test data.

A.2.1 Perplexity on Test Data of the Same Style

style	parsing	n-grams order												
		1	2	3	4	5	6	7	8	9	10	11	12	13
renaissance	rat0	113.88	27.60	20.01	19.86	19.93	20.71	20.89	21.27	21.34	19.89	19.86	19.86	19.85
	rat0clust	82.16	23.41	16.91	16.66	16.68	17.34	17.47	17.76	17.80	16.44	16.43	16.39	16.38
	rat0mod	81.64	23.25	16.79	16.55	16.57	17.22	17.35	17.64	17.67	16.32	16.31	16.27	16.26
baroque	rat0	105.22	39.25	23.24	19.69	18.55	18.65	18.66	18.95	19.01	16.80	16.80	16.87	16.82
	rat0clust	97.98	37.11	22.08	18.63	17.52	17.61	17.62	17.90	17.95	15.88	15.88	15.94	15.90
	rat0mod	94.49	36.13	21.53	18.13	17.04	17.11	17.11	17.38	17.43	15.42	15.42	15.48	15.44
classicism	rat0	117.43	42.11	26.26	22.74	21.62	21.65	21.54	21.78	21.75	19.35	19.32	19.35	19.28
	rat0clust	109.81	39.29	24.51	21.16	20.11	20.14	20.05	20.27	20.25	17.96	17.92	17.94	17.88
	rat0mod	99.61	37.64	23.19	20.04	18.98	19.00	18.93	19.13	19.11	16.93	16.90	16.91	16.85
romanticism	rat0	131.54	52.71	36.32	34.18	32.95	33.18	33.27	33.60	33.49	31.94	31.93	32.03	31.95
	rat0clust	120.57	48.81	33.86	31.71	30.52	30.72	30.80	31.10	30.99	29.54	29.53	29.63	29.55
	rat0mod	109.71	45.33	31.43	29.41	28.22	28.39	28.52	28.81	28.73	27.41	27.39	27.49	27.42

style	parsing	n-grams order												
		1	2	3	4	5	6	7	8	9	10	11	12	13
	rat0	126.55	85.96	71.27	69.52	71.16	71.59	71.75	72.17	72.45	70.45	70.59	70.66	70.73
	rat0clust	126.47	86.78	72.51	70.76	72.40	72.93	73.07	73.47	73.79	71.76	71.90	72.00	72.08
impressionism	rat0mod	106.87	71.27	59.65	58.67	60.14	60.26	60.35	60.64	60.87	59.24	59.37	59.47	59.53
	rat0	118.17	60.77	45.18	42.19	41.51	41.80	42.01	42.33	42.44	40.86	40.96	41.10	41.17
	rat0clust	109.56	54.98	41.03	38.23	37.61	37.88	38.08	38.38	38.49	37.04	37.14	37.26	37.32
20thcentury	rat0mod	101.05	50.80	37.99	35.38	34.76	34.90	35.12	35.39	35.52	34.18	34.27	34.37	34.43
	rat0	124.35	48.59	30.54	26.24	24.58	24.58	24.48	24.74	24.73	22.74	22.73	22.81	22.78
	rat0clust	112.89	44.65	28.29	24.20	22.62	22.61	22.51	22.76	22.75	20.90	20.89	20.96	20.94
classical	rat0mod	104.95	42.37	26.78	22.88	21.33	21.31	21.24	21.47	21.47	19.72	19.70	19.78	19.75
	rat0	137.31	100.33	122.11	133.01	130.30	129.22	128.80	128.53	128.15	122.43	122.33	122.25	121.95
	rat0clust	111.61	74.66	91.27	102.10	99.00	97.99	97.68	97.49	97.21	92.61	92.53	92.47	92.24
jazz	rat0mod	108.55	154.54	148.81	166.42	161.32	159.70	159.20	158.88	158.45	151.00	150.87	150.77	150.37
	rat0	115.81	61.94	47.48	41.19	39.40	39.14	39.04	39.12	39.14	35.55	35.59	35.70	35.68
	rat0clust	101.82	54.69	42.70	36.82	35.02	34.77	34.66	34.74	34.76	31.55	31.59	31.67	31.64
pop	rat0mod	101.16	54.34	42.53	36.67	34.88	34.63	34.53	34.61	34.62	31.43	31.46	31.55	31.52
	rat0	114.67	58.93	40.28	33.95	31.45	31.40	31.11	31.20	31.17	28.13	28.12	28.13	28.12
	rat0clust	101.43	52.49	36.35	30.26	27.92	27.87	27.60	27.66	27.63	24.85	24.84	24.84	24.84
rock	rat0mod	99.59	51.97	35.98	29.92	27.62	27.55	27.28	27.35	27.31	24.58	24.56	24.57	24.57
	rat0	115.88	59.92	43.24	36.29	33.64	33.40	33.09	33.14	33.09	29.92	29.92	29.96	29.94
	rat0clust	102.00	53.10	39.02	32.50	29.94	29.70	29.40	29.44	29.39	26.50	26.49	26.52	26.51
popular	rat0mod	100.54	52.40	38.54	32.08	29.55	29.30	29.01	29.05	29.01	26.15	26.15	26.18	26.16
	rat0	127.79	70.61	53.59	43.75	40.29	39.84	39.48	39.46	39.43	34.66	34.68	34.69	34.65
	rat0clust	109.82	60.23	46.42	37.79	34.52	34.11	33.78	33.77	33.74	29.52	29.53	29.53	29.50
training	rat0mod	105.59	56.49	44.46	36.20	33.06	32.65	32.33	32.34	32.31	28.23	28.24	28.25	28.22
	rat0	132.69	58.86	39.06	32.69	29.76	29.29	28.98	29.10	29.04	26.55	26.51	26.54	26.50
	rat0clust	117.45	52.05	34.93	29.28	26.50	26.05	25.76	25.87	25.82	23.57	23.52	23.56	23.52
all	rat0mod	110.55	50.20	33.58	28.15	25.43	24.98	24.71	24.82	24.77	22.59	22.55	22.58	22.54

A.2.2 Perplexity Measured on Test Data of Different Styles

training set		renaissance	baroque	classicism	romanticism	impressionism	20thcentury	classical	jazz	pop	rock	popular	training	all
	rat0	19.89	114.08	143.97	194.59	345.86	202.04	141.51	388.41	216.52	210.21	212.62	338.52	182.13
	rat0clust	16.44	110.39	145.84	193.58	374.78	203.34	139.55	326.00	193.11	196.87	195.42	301.80	173.68
renaissance	rat0mod	16.32	108.25	135.87	181.78	322.93	198.75	133.99	324.80	192.99	195.66	194.63	296.12	168.30
	rat0	50.46	16.81	53.30	74.90	156.14	93.28	40.38	277.06	145.02	150.41	148.31	192.98	67.71
	rat0clust	36.48	15.89	49.02	67.48	144.26	84.49	37.06	196.95	119.61	125.08	122.95	148.83	58.77
baroque	rat0mod	35.94	15.43	44.40	61.37	108.19	77.20	34.51	189.54	117.22	121.13	119.61	142.32	55.32
	rat0	117.17	63.12	19.36	69.75	144.61	86.27	56.38	253.89	146.76	150.04	148.77	174.26	82.75
	rat0clust	51.81	56.19	17.96	62.02	125.78	76.14	49.45	177.10	116.26	122.75	120.21	131.92	69.21
classicism	rat0mod	51.09	53.86	16.94	57.04	100.42	69.67	46.34	170.24	113.35	118.52	116.51	126.58	65.44
	rat0	87.86	60.05	48.03	31.95	108.13	71.69	52.20	231.15	134.18	142.94	139.51	153.93	76.03
	rat0clust	51.83	54.57	43.02	29.56	94.94	63.26	46.68	164.30	108.98	118.31	114.64	119.21	64.87
romanticism	rat0mod	50.43	51.83	39.77	27.42	78.38	57.81	43.53	156.52	105.11	112.89	109.84	112.49	60.85
	rat0	196.73	106.77	90.96	104.67	70.48	100.52	103.16	299.78	195.60	193.54	194.33	225.88	134.23
	rat0clust	93.40	99.26	85.25	96.41	71.78	95.11	94.37	241.68	171.72	174.49	173.42	194.07	120.32
impressionism	rat0mod	90.10	94.07	79.04	87.57	59.25	86.84	87.49	224.56	161.10	163.92	162.83	177.25	111.43
	rat0	89.88	60.42	51.61	60.81	98.85	40.87	57.27	206.86	124.86	130.18	128.12	142.98	78.66
	rat0clust	48.48	54.62	46.97	53.91	87.76	37.05	50.81	148.39	100.84	110.45	106.66	112.71	67.06
20thcentury	rat0mod	47.37	52.00	43.14	49.42	70.85	34.18	47.26	141.21	97.38	104.88	101.94	106.36	62.79
	rat0	21.69	16.50	18.27	30.05	68.11	40.49	22.76	199.28	113.40	119.91	117.37	131.06	41.38
	rat0clust	18.42	15.49	16.86	27.51	59.01	35.94	20.92	139.02	92.41	98.92	96.37	100.72	35.85
classical	rat0mod	18.13	14.99	15.83	25.49	47.97	33.25	19.73	133.66	89.89	95.48	93.30	96.20	34.02

training set		renaissance	baroque	classicism	romanticism	impressionism	20thcentury	classical	jazz	pop	rock	popular	training	all
	rat0	261.90	207.43	187.95	229.34	413.04	211.80	214.38	123.00	180.48	227.71	208.26	185.08	200.84
	rat0clust	124.72	208.50	189.76	224.25	483.03	212.70	210.38	92.98	153.78	205.03	183.60	154.52	186.53
jazz	rat0mod	218.52	394.06	335.47	421.46	841.36	397.15	389.95	151.69	276.29	391.58	342.54	275.37	340.75
	rat0	114.44	113.91	110.25	140.14	240.95	132.89	123.63	172.82	35.55	102.87	68.38	128.40	119.98
	rat0clust	65.36	108.53	108.23	135.15	253.71	129.28	117.62	132.92	31.55	89.14	59.83	105.56	108.94
pop	rat0mod	64.89	102.86	93.03	118.07	180.58	117.02	106.17	128.80	31.43	87.07	58.89	100.54	100.57
	rat0	124.79	110.08	94.96	131.36	212.81	123.67	115.75	186.23	96.17	28.13	45.07	134.32	111.52
	rat0clust	62.00	100.32	88.71	120.10	207.04	115.28	104.65	138.54	80.39	24.85	38.99	107.99	97.19
rock	rat0mod	61.96	95.95	82.80	107.40	155.56	104.92	96.98	135.62	80.19	24.58	38.69	103.83	91.59
	rat0	97.19	101.28	90.48	123.78	217.84	118.47	108.11	170.65	35.00	27.14	29.92	118.67	99.64
	rat0clust	54.98	93.01	84.27	111.83	207.88	109.63	97.95	126.07	30.87	24.11	26.51	95.43	87.13
popular	rat0mod	54.72	88.37	77.19	98.94	154.54	98.44	89.81	121.78	30.73	23.66	26.16	90.77	81.26
	rat0	104.98	102.91	87.18	111.15	229.68	121.08	106.06	164.66	83.34	106.82	97.11	34.68	90.37
	rat0clust	58.64	96.39	82.01	102.22	214.99	112.19	97.35	122.94	70.62	92.68	83.50	29.54	80.27
training	rat0mod	57.66	92.90	73.67	92.35	162.78	103.58	90.21	120.07	69.10	90.58	81.64	28.24	75.43
	rat0	20.27	16.01	17.44	29.06	68.25	39.33	22.00	97.35	31.08	28.88	29.70	31.02	26.57
	rat0clust	17.48	15.07	16.16	26.68	59.39	35.05	20.29	72.77	26.56	25.24	25.74	25.61	23.58
all	rat0mod	17.33	14.67	15.30	24.78	48.09	32.65	19.24	70.95	26.31	24.95	25.46	25.00	22.60

A.3 Detailed Evaluation of the AME system on the MIREX05 Development Set

A.3.1 Hypotheses Search without a Melody Model

Peak Detection

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.609	0.001	0.028	0.284
train02.wav	0.402	0.000	0.010	0.332
train03.wav	0.490	0.000	0.008	0.396
train04.wav	0.480	0.000	0.031	0.255
train05.wav	0.548	0.000	0.005	0.292
train06.wav	0.565	0.000	0.010	0.489
train07.wav	0.387	0.000	0.005	0.332
train08.wav	0.469	0.000	0.014	0.291
train09.wav	0.548	0.001	0.012	0.177
train10.wav	0.155	0.000	0.000	0.009
train11.wav	0.004	0.000	0.000	0.000
train12.wav	0.152	0.000	0.013	0.000
train13.wav	0.110	0.000	0.008	0.000
average	0.378	0.000	0.011	0.220

Harmonic Spectrum Product

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.361	0.000	0.000	0.348
train02.wav	0.338	0.000	0.000	0.338
train03.wav	0.379	0.000	0.000	0.379
train04.wav	0.343	0.000	0.000	0.335
train05.wav	0.336	0.000	0.003	0.319
train06.wav	0.524	0.028	0.028	0.523
train07.wav	0.341	0.000	0.000	0.323
train08.wav	0.314	0.000	0.003	0.295
train09.wav	0.283	0.000	0.000	0.235
train10.wav	0.009	0.000	0.000	0.009
train11.wav	0.073	0.000	0.000	0.000
train12.wav	0.030	0.000	0.000	0.000
train13.wav	0.010	0.000	0.000	0.000
average	0.257	0.002	0.003	0.239

Cepstrum-Biased Harmonic Spectrum Product

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.558	0.249	0.315	0.381
train02.wav	0.384	0.038	0.062	0.273
train03.wav	0.538	0.290	0.292	0.480

train04.wav	0.425	0.151	0.177	0.252
train05.wav	0.394	0.252	0.285	0.282
train06.wav	0.529	0.148	0.196	0.401
train07.wav	0.405	0.101	0.125	0.287
train08.wav	0.471	0.173	0.195	0.299
train09.wav	0.542	0.451	0.461	0.448
train10.wav	0.356	0.087	0.093	0.088
train11.wav	0.316	0.157	0.158	0.157
train12.wav	0.492	0.030	0.089	0.030
train13.wav	0.473	0.049	0.106	0.049
average	0.453	0.167	0.196	0.264

Salience Function

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.552	0.484	0.491	0.445
train02.wav	0.518	0.303	0.312	0.326
train03.wav	0.547	0.172	0.184	0.276
train04.wav	0.482	0.137	0.204	0.179
train05.wav	0.522	0.495	0.513	0.389
train06.wav	0.478	0.138	0.148	0.254
train07.wav	0.519	0.273	0.277	0.303
train08.wav	0.591	0.209	0.300	0.240
train09.wav	0.530	0.206	0.248	0.204
train10.wav	0.715	0.209	0.209	0.209
train11.wav	0.590	0.148	0.217	0.148
train12.wav	0.732	0.176	0.208	0.176
train13.wav	0.611	0.056	0.131	0.056
average	0.568	0.231	0.265	0.247

CBHSP and Salience Function Combination

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.510	0.255	0.258	0.460
train02.wav	0.382	0.073	0.076	0.327
train03.wav	0.464	0.176	0.176	0.414
train04.wav	0.439	0.149	0.161	0.330
train05.wav	0.497	0.309	0.309	0.460
train06.wav	0.537	0.117	0.117	0.492
train07.wav	0.537	0.187	0.197	0.404
train08.wav	0.354	0.095	0.106	0.272
train09.wav	0.330	0.218	0.229	0.241
train10.wav	0.302	0.133	0.133	0.141
train11.wav	0.270	0.124	0.128	0.124
train12.wav	0.180	0.002	0.002	0.002
train13.wav	0.285	0.023	0.030	0.023
average	0.391	0.143	0.148	0.284

A.3.2 Hypotheses Search with Raw Melody Model

Peak Detection

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.624	0.015	0.055	0.289
train02.wav	0.418	0.000	0.007	0.332
train03.wav	0.500	0.005	0.005	0.399
train04.wav	0.496	0.007	0.026	0.263
train05.wav	0.564	0.004	0.012	0.299
train06.wav	0.536	0.000	0.008	0.452
train07.wav	0.404	0.000	0.002	0.324
train08.wav	0.467	0.022	0.033	0.306
train09.wav	0.537	0.000	0.048	0.178
train10.wav	0.182	0.000	0.000	0.009
train11.wav	0.015	0.000	0.000	0.000
train12.wav	0.208	0.028	0.036	0.028
train13.wav	0.137	0.000	0.008	0.000
average	0.391	0.006	0.019	0.221

Harmonic Spectrum Product

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.459	0.003	0.038	0.346
train02.wav	0.338	0.000	0.000	0.338
train03.wav	0.357	0.000	0.001	0.300
train04.wav	0.368	0.003	0.005	0.330
train05.wav	0.354	0.046	0.063	0.269
train06.wav	0.524	0.000	0.000	0.509
train07.wav	0.339	0.000	0.000	0.308
train08.wav	0.300	0.000	0.000	0.268
train09.wav	0.350	0.000	0.010	0.188
train10.wav	0.009	0.000	0.000	0.009
train11.wav	0.109	0.000	0.000	0.000
train12.wav	0.055	0.000	0.000	0.000
train13.wav	0.035	0.010	0.021	0.010
average	0.277	0.005	0.011	0.221

Cepstrum-Biased Harmonic Spectrum Product

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.567	0.085	0.228	0.265
train02.wav	0.417	0.055	0.071	0.285
train03.wav	0.554	0.140	0.143	0.366
train04.wav	0.449	0.064	0.096	0.194
train05.wav	0.461	0.086	0.128	0.172
train06.wav	0.482	0.079	0.128	0.309
train07.wav	0.407	0.063	0.085	0.255

train08.wav	0.495	0.051	0.101	0.220
train09.wav	0.587	0.202	0.245	0.252
train10.wav	0.365	0.020	0.044	0.022
train11.wav	0.443	0.021	0.113	0.021
train12.wav	0.546	0.009	0.025	0.009
train13.wav	0.523	0.007	0.051	0.007
average	0.484	0.068	0.112	0.183

Salience Function

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.576	0.245	0.295	0.273
train02.wav	0.544	0.101	0.157	0.194
train03.wav	0.533	0.129	0.138	0.217
train04.wav	0.483	0.169	0.180	0.208
train05.wav	0.576	0.076	0.158	0.117
train06.wav	0.524	0.056	0.077	0.213
train07.wav	0.533	0.098	0.151	0.177
train08.wav	0.607	0.184	0.221	0.247
train09.wav	0.571	0.140	0.189	0.159
train10.wav	0.736	0.037	0.093	0.038
train11.wav	0.692	0.080	0.205	0.080
train12.wav	0.695	0.004	0.080	0.004
train13.wav	0.743	0.064	0.110	0.064
average	0.601	0.106	0.158	0.153

CBHSP and Salience Function Combination

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.532	0.109	0.140	0.365
train02.wav	0.460	0.032	0.057	0.281
train03.wav	0.492	0.091	0.103	0.325
train04.wav	0.434	0.104	0.117	0.302
train05.wav	0.524	0.245	0.245	0.385
train06.wav	0.512	0.111	0.125	0.424
train07.wav	0.564	0.090	0.136	0.329
train08.wav	0.392	0.020	0.055	0.221
train09.wav	0.378	0.120	0.151	0.162
train10.wav	0.370	0.027	0.037	0.037
train11.wav	0.312	0.111	0.126	0.111
train12.wav	0.242	0.010	0.012	0.010
train13.wav	0.521	0.085	0.090	0.085
average	0.441	0.089	0.107	0.234

A.3.3 Hypotheses Search with Clustered Melody Model

Peak Detection

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.628	0.001	0.009	0.279
train02.wav	0.418	0.000	0.003	0.332
train03.wav	0.504	0.001	0.004	0.396
train04.wav	0.502	0.009	0.019	0.264
train05.wav	0.562	0.004	0.017	0.299
train06.wav	0.550	0.000	0.011	0.452
train07.wav	0.404	0.004	0.008	0.327
train08.wav	0.467	0.040	0.042	0.319
train09.wav	0.557	0.009	0.018	0.186
train10.wav	0.182	0.000	0.000	0.009
train11.wav	0.015	0.000	0.000	0.000
train12.wav	0.208	0.003	0.010	0.003
train13.wav	0.137	0.000	0.011	0.000
average	0.395	0.006	0.012	0.221

Harmonic Spectrum Product

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.459	0.053	0.083	0.379
train02.wav	0.338	0.000	0.000	0.338
train03.wav	0.357	0.000	0.001	0.300
train04.wav	0.368	0.000	0.003	0.328
train05.wav	0.354	0.014	0.027	0.248
train06.wav	0.524	0.000	0.000	0.509
train07.wav	0.339	0.000	0.000	0.308
train08.wav	0.300	0.000	0.003	0.268
train09.wav	0.350	0.001	0.021	0.188
train10.wav	0.009	0.000	0.000	0.009
train11.wav	0.109	0.000	0.000	0.000
train12.wav	0.055	0.000	0.000	0.000
train13.wav	0.035	0.000	0.011	0.000
average	0.277	0.005	0.011	0.221

Cepstrum-Biased Harmonic Spectrum Product

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.567	0.105	0.189	0.278
train02.wav	0.417	0.032	0.051	0.269
train03.wav	0.554	0.153	0.173	0.373
train04.wav	0.449	0.049	0.106	0.184
train05.wav	0.461	0.050	0.100	0.149
train06.wav	0.482	0.014	0.038	0.277
train07.wav	0.407	0.043	0.072	0.241

train08.wav	0.501	0.040	0.087	0.206
train09.wav	0.587	0.126	0.173	0.194
train10.wav	0.365	0.036	0.054	0.038
train11.wav	0.443	0.059	0.061	0.059
train12.wav	0.557	0.002	0.057	0.002
train13.wav	0.523	0.068	0.104	0.068
average	0.486	0.060	0.097	0.180

Salience Function

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.576	0.100	0.156	0.178
train02.wav	0.544	0.103	0.166	0.196
train03.wav	0.536	0.045	0.096	0.168
train04.wav	0.486	0.051	0.112	0.130
train05.wav	0.573	0.064	0.213	0.104
train06.wav	0.520	0.020	0.079	0.191
train07.wav	0.528	0.041	0.125	0.140
train08.wav	0.591	0.075	0.149	0.156
train09.wav	0.571	0.080	0.139	0.105
train10.wav	0.755	0.035	0.073	0.037
train11.wav	0.746	0.061	0.083	0.061
train12.wav	0.715	0.051	0.163	0.051
train13.wav	0.741	0.003	0.034	0.003
average	0.606	0.056	0.122	0.117

CBHSP and Salience Function Combination

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.532	0.090	0.157	0.352
train02.wav	0.460	0.012	0.037	0.268
train03.wav	0.492	0.026	0.038	0.286
train04.wav	0.445	0.075	0.110	0.280
train05.wav	0.524	0.060	0.139	0.263
train06.wav	0.512	0.000	0.039	0.370
train07.wav	0.564	0.051	0.115	0.303
train08.wav	0.392	0.007	0.047	0.212
train09.wav	0.378	0.016	0.092	0.083
train10.wav	0.370	0.017	0.036	0.026
train11.wav	0.290	0.072	0.094	0.072
train12.wav	0.242	0.010	0.012	0.010
train13.wav	0.521	0.048	0.049	0.048
average	0.440	0.037	0.074	0.198

A.3.4 Hypotheses Search Using only the Raw Melody Model

Peak Detection

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.626	0.009	0.050	0.285
train02.wav	0.418	0.000	0.007	0.332
train03.wav	0.501	0.005	0.005	0.399
train04.wav	0.495	0.007	0.026	0.263
train05.wav	0.564	0.004	0.012	0.299
train06.wav	0.536	0.000	0.000	0.452
train07.wav	0.404	0.001	0.002	0.325
train08.wav	0.468	0.021	0.031	0.306
train09.wav	0.537	0.000	0.045	0.178
train10.wav	0.182	0.000	0.000	0.009
train11.wav	0.015	0.000	0.000	0.000
train12.wav	0.208	0.031	0.040	0.031
train13.wav	0.137	0.000	0.008	0.000
average	0.392	0.006	0.018	0.221

Harmonic Spectrum Product

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.459	0.031	0.066	0.364
train02.wav	0.338	0.000	0.000	0.338
train03.wav	0.357	0.002	0.002	0.301
train04.wav	0.368	0.003	0.005	0.330
train05.wav	0.354	0.002	0.017	0.241
train06.wav	0.524	0.000	0.000	0.509
train07.wav	0.339	0.000	0.000	0.308
train08.wav	0.300	0.000	0.000	0.268
train09.wav	0.350	0.011	0.038	0.196
train10.wav	0.009	0.000	0.000	0.009
train11.wav	0.109	0.000	0.000	0.000
train12.wav	0.055	0.000	0.000	0.000
train13.wav	0.035	0.013	0.013	0.013
average	0.277	0.005	0.011	0.221

Cepstrum-Biased Harmonic Spectrum Product

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.567	0.083	0.230	0.263
train02.wav	0.417	0.055	0.075	0.285
train03.wav	0.554	0.115	0.126	0.351
train04.wav	0.449	0.064	0.097	0.194
train05.wav	0.461	0.075	0.126	0.165
train06.wav	0.482	0.079	0.128	0.309

train07.wav	0.407	0.079	0.100	0.265
train08.wav	0.495	0.076	0.107	0.237
train09.wav	0.587	0.200	0.245	0.251
train10.wav	0.365	0.020	0.044	0.022
train11.wav	0.443	0.021	0.113	0.021
train12.wav	0.546	0.001	0.042	0.001
train13.wav	0.523	0.007	0.051	0.007
average	0.484	0.067	0.114	0.182

Salience Function

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.576	0.252	0.302	0.277
train02.wav	0.544	0.109	0.155	0.199
train03.wav	0.533	0.128	0.137	0.217
train04.wav	0.483	0.191	0.201	0.223
train05.wav	0.576	0.054	0.137	0.103
train06.wav	0.528	0.062	0.063	0.220
train07.wav	0.533	0.098	0.151	0.177
train08.wav	0.607	0.126	0.163	0.208
train09.wav	0.569	0.118	0.157	0.141
train10.wav	0.736	0.037	0.093	0.038
train11.wav	0.692	0.078	0.200	0.078
train12.wav	0.695	0.004	0.093	0.004
train13.wav	0.741	0.098	0.106	0.098
average	0.601	0.104	0.151	0.153

CBHSP and Salience Function Combination

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.532	0.119	0.165	0.372
train02.wav	0.460	0.038	0.060	0.285
train03.wav	0.492	0.092	0.097	0.325
train04.wav	0.434	0.099	0.115	0.299
train05.wav	0.524	0.128	0.174	0.307
train06.wav	0.512	0.111	0.125	0.424
train07.wav	0.564	0.105	0.154	0.339
train08.wav	0.392	0.017	0.049	0.219
train09.wav	0.378	0.120	0.151	0.162
train10.wav	0.370	0.072	0.082	0.081
train11.wav	0.312	0.109	0.112	0.109
train12.wav	0.242	0.010	0.012	0.010
train13.wav	0.521	0.043	0.048	0.043
average	0.441	0.082	0.103	0.229

A.3.5 Hypotheses Search Using only the Clustered Melody Model

Peak Detection

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.630	0.040	0.054	0.304
train02.wav	0.418	0.000	0.011	0.332
train03.wav	0.518	0.001	0.012	0.396
train04.wav	0.494	0.005	0.032	0.259
train05.wav	0.563	0.035	0.040	0.319
train06.wav	0.536	0.000	0.014	0.452
train07.wav	0.404	0.000	0.004	0.324
train08.wav	0.468	0.030	0.035	0.312
train09.wav	0.543	0.005	0.042	0.185
train10.wav	0.182	0.006	0.006	0.015
train11.wav	0.015	0.000	0.000	0.000
train12.wav	0.215	0.023	0.035	0.023
train13.wav	0.137	0.006	0.006	0.006
average	0.394	0.012	0.022	0.225

Harmonic Spectrum Product

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.459	0.000	0.058	0.344
train02.wav	0.338	0.000	0.000	0.338
train03.wav	0.357	0.000	0.001	0.300
train04.wav	0.368	0.000	0.003	0.328
train05.wav	0.354	0.002	0.015	0.241
train06.wav	0.524	0.000	0.000	0.509
train07.wav	0.339	0.000	0.000	0.308
train08.wav	0.300	0.000	0.000	0.268
train09.wav	0.350	0.001	0.004	0.188
train10.wav	0.009	0.000	0.000	0.009
train11.wav	0.109	0.000	0.000	0.000
train12.wav	0.055	0.000	0.000	0.000
train13.wav	0.035	0.000	0.035	0.000
average	0.277	0.000	0.009	0.218

Cepstrum-Biased Harmonic Spectrum Product

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.567	0.183	0.239	0.328
train02.wav	0.417	0.036	0.060	0.272
train03.wav	0.554	0.117	0.146	0.352
train04.wav	0.449	0.053	0.089	0.187
train05.wav	0.461	0.031	0.096	0.136
train06.wav	0.482	0.071	0.088	0.305

train07.wav	0.407	0.065	0.094	0.255
train08.wav	0.501	0.041	0.089	0.207
train09.wav	0.587	0.017	0.077	0.111
train10.wav	0.365	0.014	0.050	0.016
train11.wav	0.443	0.070	0.108	0.070
train12.wav	0.557	0.007	0.062	0.007
train13.wav	0.523	0.093	0.116	0.093
average	0.486	0.061	0.101	0.180

Salience Function

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.576	0.061	0.164	0.153
train02.wav	0.544	0.088	0.140	0.185
train03.wav	0.550	0.116	0.145	0.208
train04.wav	0.494	0.076	0.126	0.138
train05.wav	0.575	0.057	0.147	0.104
train06.wav	0.523	0.069	0.108	0.218
train07.wav	0.533	0.030	0.106	0.132
train08.wav	0.603	0.041	0.138	0.143
train09.wav	0.563	0.086	0.171	0.117
train10.wav	0.754	0.073	0.136	0.075
train11.wav	0.692	0.019	0.163	0.019
train12.wav	0.723	0.083	0.149	0.083
train13.wav	0.743	0.019	0.048	0.019
average	0.606	0.063	0.134	0.123

CBHSP and Salience Function Combination

file	voicing accuracy	raw pitch accuracy	raw chroma accuracy	overall accuracy
train01.wav	0.532	0.079	0.131	0.345
train02.wav	0.460	0.039	0.060	0.286
train03.wav	0.492	0.125	0.135	0.345
train04.wav	0.434	0.023	0.054	0.248
train05.wav	0.545	0.053	0.177	0.258
train06.wav	0.512	0.045	0.111	0.392
train07.wav	0.564	0.051	0.126	0.303
train08.wav	0.392	0.031	0.069	0.229
train09.wav	0.378	0.048	0.087	0.107
train10.wav	0.370	0.059	0.091	0.068
train11.wav	0.290	0.013	0.017	0.013
train12.wav	0.242	0.015	0.015	0.015
train13.wav	0.521	0.024	0.069	0.024
average	0.441	0.047	0.088	0.203

B. Manual for the AME System

The AME system enclosed can be launched on a machine where installed Java SE is installed in version 6 or higher.

If you stand in the directory with the executable file, the program can be run using the following command:

```
java -jar rozehra.jar [wav file]
```

where `[wav file]` is path to the wave file containing the audio signal to be analyzed. We recommend using monophonic signal with sampling frequency of 44,100 Hz. The algorithm run is visualized in a new window. There are several options to set up the algorithm properties.

Pith Detection Algorithms

The pith detection algorithm can be selected by option `-pitch=[algorithm]` and it can have the following values:

- `peaks` – simple peak detection
- `hsp` – harmonic spectrum product
- `cbhsp` – cepstrum biased harmonic spectrum product
- `salience` – salience function maximization
- `combined` – combination of CBHSP and salience function maximization

While using the salience function maximization the program may have bigger memory requirements. Therefore we recommend bigger heap space for the Java Virtual Machine. It can be done by using option `-Xmx6g`.

If the option is not specified salience function maximization is used.

Melody Modeling

The melody model can be specified using option `-model[format]` where `format` can have following values:

- `plain` – raw n-gram model
- `clust` – n-gram model with additional clustering

If the option is not specified, no model is used.

The SRILM toolkit server process must be running while using the melody models. For that the SRILM toolkit must be installed on the machine. The process can be then run by command:

```
ngram -server-port [port number] -order 10 -lm [model file]
```

The raw melody model is in file `all-rat0.lm.gz` and the clustered model in file `all-rat0clust.lm.gz` in the directory `melody-models` on the enclosed DVD.

The server port number must be set also for the AME process. It can be using option `-srilmPort=[port number]`. If the `ngram` program from the SRILM toolking is not on the system path, the full path to the program can be specified using option `-ngramPath=[path]`.