N-GRAM BASED APPROACH TO COMPOSER RECOGNITION

JACEK WOŁKOWICZ, ZBIGNIEW KULKA

Institute of Radioelectronics, Warsaw University of Technology
J.Wolkowicz@elka.pw.edu.pl

The paper describes the application of the tools provided by Natural Language Processing and Information Retrieval to the music. A method of converting complex musical structure to the features (n-grams) corresponding with words for text was introduced. A mutual correspondence between both representations was shown by demonstrating certain important regularities known from the text processing, which are fulfilled also for the music. The theoretic aspects of the case were applied to the problem of automatic composer attribution where the statistical analysis of n-gram profiles, known from statistical NLP, was used. A MIDI files corpus of piano pieces was chosen as a source of data.

Keywords: Composer Recognition, Music Processing, Music Information Retrieval, N-grams

1. Introduction

Music content processing becomes an important domain of research. A lot of works regarding these tasks were published. It results from the fact, that there are more and more repositories of a musical content available on the web accessible for everybody. Meanwhile, the tools for searching and browsing the text content, such as Google, were developed and are widely used. These tools were founded on the basis of IR (Information Retrieval) and NLP (Natural Language Processing). While it was found that people started to create music concurrently with language development, one can assume that music is also a natural language with all repercussions. This implies that some techniques with proven effectiveness in NLP and IR can be applied to the music as well. We want to introduce a novel statistical approach to music analysis based on n-grams. The aim of the paper is to show, that music is as far similar to natural languages as one can process the music using the methods already developed.

One distinguishes certain levels of a text processing, listed in the Table 1. NLP, as well as music processing, tries to convey through all levels, from recording to understanding. Of course, there is no such tool that does everything at a time, i.e., understands the meaning and gets knowledge from a raw waveform. In fact, NLP tools concentrate on a certain level trying to move the problem up a level. Music, similarly to a natural language, can be recorded and presented primarily as a waveform. On the 'phonetics' level one tries to investigate the structure of a sound, separate and distinguish between notes and instruments. This task combined with notes recognition is a well known problem of contemporary sound engineers even if they do not know that they are involved in NLP tasks.

Table 1. Levels of NLP. Text vs. music.

	Text processing	Music processing
phonetics	Voice recorded	Recording
phonology	Phonemes of the language	Separated notes
morphology	Words structure	Notes in the score
syntax	Words order	N-grams, notes order
semantics	Words meaning, POS	Harmonic functions
pragmatics	The meaning of a sentence	Phrase structure
discourse	Context of a text	Piece's interpretation

The music content analysis is the next step of this so called MLP (Music Language Processing). Music has a hidden structure, hidden rules, like grammar for text. In this case it is called the harmony. It rules how to put words (notes) together, how to build phrases with them, which are well-formed. It also manages the musical meaning of the piece which is the order of chords. In the first case we can talk about the syntax of the music while in the second case – about the semantics of the certain chord or a pragmatics of a phrase. MIR (Music Information Retrieval) tools should work mainly on those levels. The other problem with the music is that there are no word boundaries and phrasing is driven by harmony, so one has to figure out the structure of the piece as well as its harmonic representation.

2. Types of musical data

There are two, much different types of musical data, that can be stored on computers.

1. Raw – the sound recorded by the microphones, compressed (e.g., mp3 format) or stored explicité (pcm files e.g, wav format).
2. Symbolic representation – score notations (mus, sib, abc, xml) or MIDI protocol.

People got used to raw representations because they like to hear "real" artists' music, not the symbolic version, which is played on every machine differently.

Same happens with text: on the one hand we may store the original author's voice, on the other – textual data. However, people got used to represent text rather symbolically. This representation is easy to store, easy to edit (using editors with keyboards), easy to process and well known for most the people from childhood. The second thing is the flatness of the text – words occur one after another, there is no concurrency in the text. It is not so simple in music and this fact should be resolved before applying NLP tools to music. Next problem is that almost everyone can read, but rarely play the music. That is why people got used to raw music representation.

While MIDI files stores the symbolic data, they may behave like textual files. Nevertheless, they consist of concurrent channels and tracks, which may overlap each other. Thus, talking about correspondence to text, one has to omit these concurrencies. We decided to treat every channel separately, if only every channel represents one hand, solving the problem of parallelism in each channel separately. The resulting output looks like a crowd play where lots of people talk at the same time. In each channel we have taken the highest currently played note because, according to basic psychoacoustic knowledge, one can show that people concentrate on them.

3. Musical data representation -- N-grams extraction

N-gram is simply the 'n' consecutive letters or words. We are talking then about 'word' or 'character' n-grams. They are overlapped, i.e., each token belongs to 'n' n-grams. For instance, in the text "Music" we have 3 character 3-grams: 'Mus', 'usi' and 'sic'. N-grams are very useful in NLP in the situation, where no only words may play role, e.g., in authorship attribution, language recognition or where it is hard to separate the words. The good example of this behavior is Thai, where there is no whitespaces. In this way, Thai is especially similar to music – for us it is just a flow of characters without an order or semantics, however it still remains a natural language for Thais. If NLP tools may also be applied to this language, why they cannot be applied to the music as well (treated as a natural language).

The first step of n-gram extraction after simplifying the data from MIDI files (i.e. having linear order of notes in each track) is to find, what could represent unigrams. The simplest approach to this task is simply getting the duration or pitch as the basic feature, but this cannot bring good results. The pieces can be played in different speeds and can be transposed to any key. The features that one need to have are that they were to be key independent, so not the pitch is important, but the relative pitch to other notes. It is important, because the key does not tell us anything about the certain work, e.g., J. S. Bach wrote two sets of preludes and fugues, each fugue in each existing key in well tempered scale, thus if one does the pitch distribution analysis, we will obtain one flat, normalized. The second important feature of musical n-grams is that they should be tempo-independent. In MIDI files the duration is not given symbolically – as quarters, eights, half-notes, but in the direct way, that can be mapped to milliseconds. Every MIDI file representing the same piece, but sequenced by different people (or programs) will look a little bit different. That is why we decided to apply the relative duration counting, not the direct one. Each difference is in the logarithmic scale and is rounded to cover some random tempo fluctuations. The formula applied to each pair of notes is given as follows:

$$(P_i, T_i) = \left(p_{i+1} - p_i, round(\log_2(\frac{t_{i+1}}{t_i})) \right) \quad (1)$$

where p_i denotes the *i'ths* note pitch (in MIDI units), t_i stands for the *i'ths* note length (in ms) and (P_i,T_i) is the resulting tuple. The procedure of extracting N-grams is shown in Figure 1:

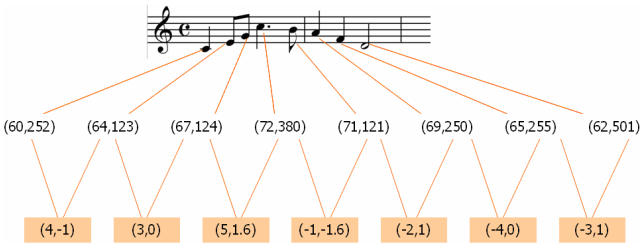

Fig. 1. Unigrams extraction.

A transition to the n-gram is simple and lies in getting n consecutive unigrams as an item. Three types of n-grams can be obtained out of this. We can consider the rhythm only, the melody only and we can also take n-grams as a combination of both these features.

N-gram representation is quite similar in its shape to text. We claim that MIR engines may be built using this representation, working on the same rule as IR engines like Google. The great work about string matching techniques, with a result of a MIR system was shown in [2]. The tool is available online, but methods still need enhancements.

4. MIDI corpus

We have collected a set of MIDI files freely available on the Internet of five different composers. We have chosen only the piano works for better compatibility. Moreover, each piece has to be well-sequenced, i.e., each channel has to represent one and only one staff or hand. The reason for this is that it is very easy to produce the MIDI sequence that sounds well, but is messy inside. The number of pieces and the size are given in the Table 2:

Table. 2. MIDI corpus properties.

Composer	Training Set	Testing Set
J. S. Bach	99 items, 890kB	10 items, 73kB
L. van Beethoven	34 items, 1029kB	10 items, 370kB
F. Chopin	48 items, 870kB	10 items, 182kB
W. A. Mozart	15 items, 357kB	2 items, 91kB
F. Schubert	18 items, 863kB	5 items, 253kB

While considering music files one has to point out, that there are big disproportions between pieces. Some miniatures are quite tiny, but there are also very large forms, like concertos. Thus, it is better to describe the volume of corpora in bytes rather than in number of pieces. The second important thing is to know that the differences between composers derive from composers' background and their lifetimes, e.g., greater difference is between F. Schubert and J. S. Bach than between F. Schubert and F. Chopin because they both lived in 19th century.

5. Zipf's law for music

There is a certain number of regularities and laws, that lie on the foundation of NLP and IR. These laws show that text is not a set of words distributed randomly. We are describing below the one, very important law – Zipf's Law, which describes the distribution of words in text. It allows estimating certain features of the IR system before building and running it.

Figure 2 was obtained for the piano pieces for the corpus described above. It shows the number of occurrences of each n-gram as a function of its rank, i.e., the place of each 'word' in the sorted frequency table in descending order. According to Zipf's law, the frequency of any word is roughly inversely proportional to its rank. If dimensions are in the logarithmic scale, the relation should be linear.

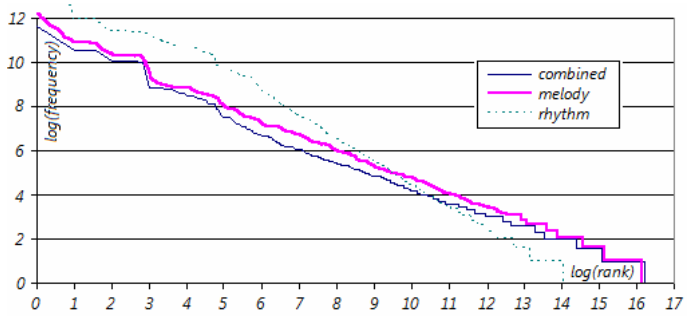

Figure. 2. Zipf's Law for music for three types of n-grams.

Despite some irregularities at the beginning, the law is satisfied. We may notice the difference between rhythmic and melodic profiles. There are much more low-rank rhythmic n-grams than melodic ones and much more high-rank melodic n-grams than rhythmic ones. It means, that a rhythm is usually much simpler then a melody and that melody remains unique for every theme because most of the melodic patterns occur a few times.

6. Entropy analysis

We can distinguish certain groups of words in text, such as key-words, stop-words, noise-words. Key-words are the words with a meaning and important semantic value for the text. Their rank should be in the middle of logarithmic rank value. Stop-words are the most frequent words, like 'the', 'a', 'and'. They do not have any semantic meaning and usually mess up analysis. Noise-words are the words that occur few times and they do not make us come to any conclusion. The definitions of the preceding words' groups are 'semantic' so they cannot be applied to the music i.e. we cannot simply name a phrase a key-word. The notion that may help in this situation is entropy as a measure of information.

Regarding entropy, key-words are the words (n-grams) with high entropy inside the class, and small entropy among all classes. Hence, noise-words are the n-grams with high entropy in both areas while stop-words have both rates low. After counting all occurrences in each group we obtained a distribution shown in Figure 3. More details of the method may be found in [3]. One may notice, that the position of each group is as expected so the structure of music pieces corresponds to the one of text documents, which once more shows, that music can be treated as a natural language, because it is a natural language.

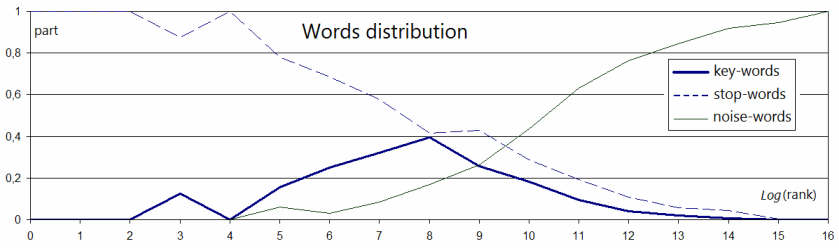

Figure 3. N-grams' distribution among corpus.

7. Composer recognition based on n-gram analysis

According to the outcome of the reasoning from the previous sections, one may use NLP tools used already for text. We decided to touch the composer recognition task, while it was shown that it may be successfully done on text, as an authorship attribution [1]. However, the use of this method on musical content is a novel approach.

Similarly to the authorship attribution, we have created a profile of each composer as a table containing n-grams with their occurrences in all pieces of each composer from training corpus. If a new piece comes to the system, the program counts all occurrences of each n-gram and creates a profile of the testing piece. The profile is then compared to the composer profiles and the most similar is taken as a result.

There are certain degrees of freedom of the system. The algorithm was testes for different n-gram sizes, profile's sizes, and aging factor during creating the primary profiles. Results for the best obtained aging factor (0.96) are shown in Table 3. The accuracy reaches 84% for the high profiles sizes and for n equal to 6. It means, that the average 'music' word has about 7 notes (6-gram describes 7 consecutive notes), which is usually a measure.

Table 3. Results of the algorithm.

n \ size	100	250	500	1000	2500	5000	10000
2	0.41	0.38	0.38	0.35	0.32	0.43	0.43
3	0.46	0.54	0.59	0.62	0.59	0.51	0.43
4	0.62	0.70	0.65	0.73	0.73	0.78	0.86
5	0.54	0.62	0.70	0.78	0.78	0.81	0.81
6	0.54	0.59	0.68	0.68	0.84	0.78	0.84
7	0.46	0.49	0.68	0.68	0.68	0.70	0.84
9	0.46	0.57	0.49	0.51	0.57	0.68	0.76
12	0.41	0.46	0.41	0.41	0.41	0.46	0.49

8. Conclusions

Our analysis shows that music can be sensible to the NLP and IR tools and some cases of this problem were proved in this paper. Using n-gram interpretation may allow to index and efficiently browse musical libraries, which can be a major problem nowadays. The usefulness of the methods were proven in the case of composer recognition, however, we claim that there is plenty of tasks that may be solved using these methods.

References

[1] KESELJ V., PENG F., CERCONE N. and THOMAS C. *N-gram-based Author Profiles for Authorship Attribution.* August 2003, In *Proceedings PACLING'03*, Halifax NS CA, pp. 255-264.

[2] LEMSTROM K. *String matching Techniques for Music Retrieval*. Ph.D. Thesis. Helsinki, April 2000.

[3] WOŁKOWICZ J., *Analysis of piano pieces corpus in MIDI files (Polish)*, Project Report, Warsaw University of Technology, 2007. Online resource. http://torch.cs.dal.ca/~jacek/papers/projects/classification_enhancement.pdf