Analysis and Prediction of Piano Performances Using Inductive Logic Programming

Erika Van Baelen and Luc De Raedt

Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
Email: Luc.DeRaedt@cs.kuleuven.ac.be

Abstract. Starting from the work of Matthew Dovey on analysing Rachmaninoff's piano performances using inductive logic programming, we show how to apply the clausal discovery engine Claudien to induce theories for predicting MIDI files from the musical analysis of a score.
This extends Dovey's work in several directions: MIDI-encodings are used instead of the older Ampico, a richer musical analysis within LaRue's SHMRG-model is applied, a much finer qualitative analysis of features is learned (making it nearly quantitative), and predictions are made.
The application is not only relevant as yet another inductive logic programming benchmark, but also as a demonstration of the need for multiple predicate learning, sequence prediction and number handling in inductive logic programming. Furthermore, the results presented here can be considered the first original application of the clausal discovery engine Claudien.

1 Motivation and Introduction

The motivation for this research comes from the paper by Matthew Dovey on *Analysis of Rachmaninoff's Piano Performances using Inductive Logic Programming* [6]. The starting point of Dovey's study were Ampico recordings of two Piano performances by Rachmaninoff in the 1920's, i.e. of Rachmaninoff's Prelude in C Sharp Minor, op. 3 no. 2 (Ampico Roll Number 57504) and of Mendelssohn's Song Without Words, op. 67 no. 4 (Ampico Roll Number 59661). Ampico recordings capture a performance of a piano player by encoding the notes, duration and tempo, as well as the dynamics of the key pressure and pedalling. Starting from Ampico recordings and knowledge about the musical structure of the piece being performed, Dovey produced two datasets (one for each piece) and analysed them using the inductive logic programming system Progol [9]. This enabled Dovey to induce some general rules underlying piano performances by Rachmaninoff. Dovey's work did not only show that music is an interesting application area for inductive logic programming, it also potentially contributes to the analysis of music.

Despite these contributions, Dovey's work was severely limited. Firstly, Dovey only used the melody notes of the pieces, which is in the case of piano rather awkward. Secondly, the rules induced were all qualitative, and hence could not be used for reconstructing (or predicting) Ampico recordings. Thirdly, the musical

analysis employed by Dovey is rather simple. Furthermore, Dovey suggested that the use of MIDI (Musical Instrument Digital Interface), which is the current information medium for encoding performances on synthesizers, could be more useful.

In this work, an attempt is made to improve the results of Dovey. To this aim, Dovey's setting was adapted in the following manner. First, instead of using the older Ampico recordings, we also use MIDI. Secondly, all notes will be taken into account. Thirdly, an extended musical analysis, using Jan LaRue's SHMRG model [7] is employed. Finally, and most importantly, rather than discovering qualitative regularities which capture some intuitions about piano performance, it was our aim to discover regularities that can be used to generate MIDI encodings starting from the musical analysis of a piece. So, we try to generate the performance data from the musical analysis of the piece, which means that the musical score (with some background information) is actually being interpreted. It is our hope that this would help to improve upon the current performances that are generated automatically by software programs.

These issues were addressed in two steps. In the first step, we started from Dovey's Ampico data (which were kindly supplied by Dovey), but extended it with new musical knowledge as well with quantitative aspects. In the second step, we used a MIDI encoding[1] (performed by Carl Verbraeken) of Mendelssohns's Lied Ohne Worte together with the musical analysis. In both steps, the first half of the data was used as a learning set. Rules were then induced by the Claudien system [2, 5] and applied on the musical analysis of the second part in order to produce an interpretation of this second part (whether MIDI or Ampico). In addition to the interpretation (on MIDI or Ampico), which represents the musical result of this work, several performance aspects were also analysed using more traditional machine learning techniques, which results in a number of statistics and graphs.

Viewed from inductive logic programming, this work represents a true application of multiple predicate learning, and hence demonstrates the relevance of this type of learning (cf. [4]). Also, it contains the first *novel* application realised with the clausal discovery engine Claudien [5], and in this respect, it shows the possibilities of the underlying data mining approach to inductive logic programming.

The paper is structured as follows: Section 2 presents the extended representation format used in the Ampico studies, Section 3, presents the representation needed for MIDI, Section 4 presents the prediction algorithm employed, Section 5 contains some results of the empirical evaluation, and finally Section 6 presents some conclusions and discusses further work.

2 Ampico recordings

As stated in the introduction, the first part of our study starts from Dovey's representation of the Ampico recordings. However, his representation was mod-

[1] There exist no original MIDI encodings of Rachmaninoff's performances.

ified significantly, the most important modification being the use of an enriched musical theory based on the SHMRG-model of Jan LaRue [7]. SHMRG stands for Sound, Harmony, Melody, Rhythm and Growth. LaRue provides recommendations for describing each of these different musical aspects. The model used by Dovey can be considered a simpler SHMRG-model, as Dovey uses this only for Rhythm, Sound and Melody. In this paper, we will employ knowledge about Sound, Harmony, Melody and Rhythm. The aspect Growth is less important as this investigation concerns a single (small) piece for piano, and not for instance a complicated symphony. In addition to also addressing Harmony, we also contribute many new features for Rhythm, Sound and Melody.

On the other hand, we did not include in our investigation more advanced musical analysis techniques such as those of Schenker and Meyer, nor models from cognitive musicology (such as Narmour and Lehrdals and Jackendoff) which are quite popular in artificial intelligence studies of music. The main reason for not applying these, is that we wished to keep the representation as simple and understandable as possible. This could however be handled in further work on this topic.

We now present in detail the descriptions we used in the SHMRG-model. While doing so, we indicate where we deviate from Dovey, and in what manner. We also assess the objectiveness of each feature.

2.1 Sound

Sound captures the tone or colour of the music. For our purposes (the use of piano only), this issue reduces to information about accentuation, force and touch of the different notes, as found on the score.

- *Touch* (source : score, changed w.r.t. Dovey) : staccato (S), legato (L), portato (P), leggiero (LG) or normal (). Dovey uses terms such pizzicato and portamento, which do not really apply to piano.
- *Accent* (source : score, slightly changed w.r.t. Dovey): sforzando (S), sforzato (>), fermate (F).
- *Expected Force* (source: score, taken from Dovey) in a range from 1 to 14.
- *Actual Force* (source: Ampico, taken from Dovey) in a range from 1 to 14.
- *Expected Cresc./Dim.* : (source : score, new parameter, subjective for what concerns the end of each cresc./dim.): cresendo (C) or diminuendo (D).

2.2 Harmony

Harmony is the study of chords and intervals, of the ways in which chords and intervals are related to one another, and the ways in which one interval or chord can be connected to another. Harmony is not really addressed by Dovey, as he treats harmony and melody alike. We apply Cope's SPEAC-model on Harmony here. The SPEAC encoding stands for Statement, Preparation, Extension, Antecedent and Consequence. It is devised according to linguistic principles. In the

SPEAC model every note is assigned one of S,P,E,A, or C, in order to characterize the function of the note. A linguistic example, due to Dovey, is the sentence 'I am going to the store', in which 'I' maps to a *Statement* (something which exists by itself), 'am going' maps to an *Antecedent* (which requests a response), 'store' maps to *Consequence* (which represents the answer to the request), and 'to the' maps to *Preparation*. Dovey applied the SPEAC model in a non-hierarchical manner to the melodic structure of the piece. Here we apply it on the harmonic structure of the piece.

This results in modified SPEAC feature for each note:

- *SPEAC*: (source: score, changed from Dovey, subjective).

2.3 Melody

This concerns the melodic theme of the work. For each note, we keep track of :

- *Phrase position* : (source : score, as for Dovey, subjective), beginning (B) end (E), or left blank for a note in the middle of a phrase.
- *Phrase shape*: (source: score, new parameter), indicates the form of the global melodic sentence, i.e. rising (R), falling (F), level (L), waveform (W), sawtooth (S), undulating (U).
- *QA*: (source : score, new parameter) which indicates (when applicable) whether two voices are in dialogue, question (Q) or answer (A).
- *Phrase stress*: (source : score, new parameter, subjective), the note with the strongest accent in a melodic sentence is marked with an S[2].

2.4 Rhythm

Rhythm is concerned with everything that concerns tempo (the global level) and metrics (at the local level).

- *Tempo density* (source: score, new, subjective) : on a scale from 1 (broad) to 5 (dense).
- *Rhythmic stress* (source: score, new, subjective) : the note with the strongest accent in a rhythmic sentece is marked with an S.
- *Expected gap*: (source: score, as Dovey) the expected gap between one note and the previous one, in 1/120 beats.
- *Actual gap* (source: Ampico, as Dovey) the actual gap between one note and the previous one, in 1/120 beats.

[2] This feature goes back to Rachmaninoff (cf. Rachmaninoff, The complete recordings, RCA, Victor Golden Seal, BMG Music, 1992.) Rachmaninoff approached the literature of the piano as only a composer could. He described his method in these words: " 'You must take the work apart, peer it into every corner, before you can assemble the whole'. He shared with his friend Feodor Chaliapin, ..., the interpretative method of determining the climax or 'point' of a piece. Once that was determined, the structure of the piece proceeded to and receded from the 'the point' with inexorable logic."

- *Expected duration*: (source: score, as Dovey) the expected duration of a note, in 1/120 beats.
- *Actual duration* (source: Ampico, as Dovey) the actual duration of a note, in 1/120 beats.
- *Pitch* (source : score, as for Dovey) the pitch of the note. C0 represented middle C, C-1, the C one octave below middle C, C1 one octave above and so on.
- *Bar* (source : score, as for Dovey) the bar number of the note.
- *Beat* (source : score, as for Dovey) the beat number within the bar of the note.
- *Offset* (source : score, as for Dovey) the offset of the note w.r.t. the previous beat.

These last four features are not employed in the learning phase, as they precisely encode what is on the score. They are irrelevant for prediction purposes in our context, though some derivatives are used, such as e.g. pitch-descending, ...

The sequence of the notes is encoded in the following predicate:

- *Pre(Note1,Note2)* (source: score, as for Dovey), which denotes that Note1 comes immediately before Note2.

2.5 Representation

The above SHMRG features are first represented in a table using a kind of attribute-value representation. For instance the first 6 notes of the Prelude in C Sharp Minor correspond to :

A-1, , >, 1,1,0,0,120,11,0,87,12,P,B,F,A,,2,
G-1#,,>,1,2,0,0,120,11,3,97,11,A,,F,A,,2,
C-1#,,>,2,1,0,0,720,11,4,402,13,C,E,F,A,S,2,
C1#,P,,3,1,60,-180,60,1,-158,65,1,S,B,W,Q,,2,
E1,P,,3,2,0,0,60,1,0,61,2,P,,W,Q,S,2,
D1#,P,,3,2,60,0,60,1,2,55,1,S,E,W,Q,,2,

where the features include Pitch, Touch, Accent, Bar, Beat, Offset, Expected Gap, Expected Duration, Expected Force, Actual Gap, Actual Duration, Actual Force, SPEAC, Phrase Position, Phrase Stress, Tempo Density, Expected Cres/Dim.

Notice however that this representation is misleading as some of the relevant relations are implicit in this table. This includes the sequence of the notes, as well as the differences between expected and actual values, which actually makes this a typical *relational* or *inductive logic programming* problem.

The above table can be automatically translated into a relational representation (cf. below for an example). Such a representation was employed by Dovey in order to predict in a qualitative manner the following properties of a note : duration, duration of the gap w.r.t. the previous note, and force [3].

[3] Dovey actually named these predicates differently.

- *duration*, which provides information about the actual duration of the Note as compared to the expected duration.
- *detach, overlap*, which provide information about the duration of the actual detach or positive gap (resp. overlap or negative gap) of a Note as compared to the expected detach (resp. overlap).
- *timing(Note)*, which indicates how the time between the start of the given note and the preceding note varies between performance and score.
- *dynamic-actual(Note), dynamic-expected(Note)* which indicate the difference in actual (resp. expected) force between this note and the previous one.
- *dynamic-force(Note)*, which indicates the difference between the actual force of this note and the expected force.

All of the above aspects were encoded qualitatively, using 3 to 5 different values for each aspect. Using this type of representation, Dovey was able to discover 75 qualitative rules capturing some of the intuitions behind Rachmaninoff's performances in these pieces.

Though we also performed experiments on the qualitative data (see [11] for more information), we found it more interesting to try to induce rules which could be employed for predictive purposes. However, this requires the use of a finer scale for representing the numeric information.

2.6 Quantitative Information

We need to model 4 different aspects: force, duration, detaches and overlaps.

Let us start by force, which is encoded on a scale from 1 to 14. A straightforward way of encoding this aspect, would employ two predicates, one for the expected force and one for the actual force, e.g. *exp-force(Note,Int)* and *act-force(Note,Int)* where Note is the number of the note, and Int a number between 1 and 14. As the relative difference is more important than the specific forces, we choose to encode this aspect using a predicate

- *dynamics-ch(Note,RelInt)* where *RelInt* can now take a value between -13 and +13, indicating the difference between the actual and expected force. E.g. when the expected force of note19 is 11 and the actual is 13, this corresponds to *dynamics-ch(note19,-2)*.

Whereas force is represented on a scale from 1 to 14 (due to the nature of the Ampico rolls), time and hence duration, detaches and overlaps are represented on a much more detailed scale (1/120 of a beat). Due to the number handling problems of current inductive logic programming systems, we did not represent these aspects in the same manner as for force. Furthermore, such a fine grained division is hardly noticeable for the human. Therefore, we used as features the actual time divided by the expected time, and discretized this into 23 different intervals. This yields the following predicates:

- *duration-factor(Note,Class)* where class takes one of the following intervals: less than 0.1, between 0.1 and 0.2, between 0.2 and 0.3, ..., between 0.8 and

0.9, between 0.9 and 0.95, between 0.95 and 1, between 1 and 1.05, between 1.05 and 1.1, between 1.1 and 1.2, ..., between 1.9 and 2, more than 2. E.g. when the expected duration of note202 is 120 units, and the actual is 20, this would be noted as *duration-factor(note202,.10-.20)*

- *detach-factor(Note,Class)* and *overlap-factor(Note,Class)* similarly, where it is assumed that expected detaches (resp. overlaps) are actual played as detaches (resp. overlaps).
- *zero-gap(Note,Class)* this is to avoid divisions by 0 in case the expected gap is 0, and is only used for small actual gaps and detaches (i.e. less than 4/120 of a beat). Here, Class takes the value ok (real gap of 0), small-detach (real gap between 1 and 4) or small-gap (real gap between -1 and -4). E.g. when the expected gap between note132 and note 133 is 0 units, and there is a gap of 4 units, this is encoded as *zero-gap(note133,small-detach)*
- *zero-detach-factor(Note,Class)* and *zero-overlap-factor(Note,Class)* are used when the expected gap is 0, and the actual gap is larger than 4/120 of a beat. Then Class denotes the actual detach (resp. overlap) divided by the duration of the note, for encoding the relative time of an overlap (detach). E.g. when the expected gap between note133 and note34 is 0 units, and there was a gap of 15 units while the expected duration of note134 is 120 units, we have *zero-detach-factor(note134,.1-.2)*

When the aim is to induce the definitions of these predicates, we are actually facing a *multiple-predicate learning* task, as these predicates are possibly mutually dependent.

3 MIDI data

MIDI stands for Musical Instrument Digital Interface, first published in 1983 by International MIDI Association (IMA). MIDI does not have the same quality of a CD, as its encoding is much simpler. The encodings are also specific for each instrument or synthesizer. W.r.t. Ampico MIDI is a significant advance, it has e.g. 128 levels to encode the force instead of 14 for Ampico, and it applies not only to piano, but to any digital instrument. However, the ideas are very similar. It is because we did not have any Ampico device available that we employed MIDI.

Our experiments with MIDI, were all carried out on a performance by Carl Verbraeken of Mendelssohn's Lied Ohne Worte. Two important changes were made w.r.t. the Ampico data. First, the data were extended to include also the non-melody notes. Just playing the melodic notes on a piano is not interesting. Second, the right pedal of the piano was taking into account. When this pedal is applied, the strings are not damped. As the left pedal was always applied in our case study, we did not do anything special for this pedal. Other changes were made so that the Ampico data format could be reused. Beats on MIDI (in our study) were counted in 384 ticks, Ampico in 120 ticks. Velocity in MIDI is connected with volume and is counted in 128 levels, Ampico's force is encoded in 14 levels. We found it convenient to reduce these 384 ticks to 120 and the

128 levels to 14. This is useful for recycling the encoding and for comparing the results. (In this respect, our work is closer to Ampico than to MIDI, despite the fact that our predictions are used to generate a MIDI file).

New predicates were defined to capture the pedal and the non-melodic notes. Otherwise the same data representation format as above was employed.

- *pedal(MelNote,State)* (source MIDI, new), where State is one of : on, off, initiated, terminated.
- *melody-note(Note)* (source : score, new), which denotes that Note is a melody note
- *related-to-melodynote(Note1,Note2,How)* (source : score, new), where Note1 is a non-melodynote, Note2 a melody note, and How is one of chord, filler note , and grace note. At present, a non-melody-note is only related to one melody note.

Otherwise the same representation is used for melody-notes as for non-melody notes.

An example of the real representation of some notes is given below:

melody-note(note328).
pedal(note328,on).
staccato(note328).
no-accent(note328).
antecedent-function(note328).
middle-phrase(note328).
melody-undulating(note328).
pitch-descending(note328).
no-melodic-stress(note328).
no-rhythmic-stress(note328).
triplet-quaver(note328).
duration-factor(note328,< .1).
exp-gap(note328,zero-gap).
zero-detach-factor(note328,.6-.7).
exp-force(note328,5).
no-cresc-or-dim(note328).
dynamics-ch(note328,-1).
pre(note326,note328).

related-to-melody-note(note329,note328,chord).
triplet-semiquaver(note329).
duration-factor(note329,.8-.9).
exp-gap(note329,zero-gap).
zero-gap(note329,small-gap).
exp-force(note329,5).
dynamics-ch(note329,1).

related-to-melody-note(note330,note328,chord).

triplet-semiquaver(note330).
duration-factor(note330,.6-.7).
exp-gap(note330,zero-gap).
zero-gap(note330,small-detach).
exp-force(note330,5).
dynamics-ch(note330,2).

4 Clausal Discovery

Using the above datasets several experiments were performed with the clausal discovery engine Claudien [5, 2], cf. [11]. First, rules were derived with a similar purpose as Dovey, i.e. to get insight in the performance characteristics of Rachmaninoff. So, the aim here was to generate understandable and intuitive regularities. This was quite succesful, however, due to space restrictions we do not further elaborate on this. Secondly, it was our aim - and this contrasts our work from that of Dovey - to be able to use the induced hypothesis in order to predict MIDI [4] performance data. To realize this, we choose to induce rules on the first half of a piece (both Mendelssohn's Lied Ohne Worte, and Rachmaninoff's Prelude in C Sharp Minor were considered) and to apply the induced hypothesis to predict the Ampico performance data of the second half of the piece. We soon discovered that the naive approach, which consists of selecting a subset of the rules produced by Claudien[5] would not work. The reason is that only few predictions are made. For instance, with Claudien's accuracy parameter set to 80 per cent, only for 3 per cent of the notes the force could be predicted, only 10 percent for duration, and 5 per cent for detaches and gaps. Lowering Claudien's accuracy parameter results in more predictions, but is not a solution. The problem is that many rules induced by Claudien are recursive and rely on the performance characteristics of the previous note(s), which is often also not known. What is needed is a number of simple non-recursive rules. We therefore used the following strategy:

1. induce rules that use the features about the current melody note itself, e.g. dynamics-ch(X,0) :- strong-accent(X), cresc-expected(X).
2. induce rules that use only the features of the previous melody note, e.g. dynamics-ch(X,0) :- pre(Y,X), staccato(Y)
3. induce rules about the previous melody note and the current melody note, e.g. dynamics-ch(X,0) :- pre(Y,X), staccato(Y), conclusion-function(X)
4. (only for MIDI) : induce rules for non-melody notes.
5. (only for MIDI) : induce rules for right pedal[6].

[4] As said before, we actually predict a kind of Ampico data, but transform it to MIDI.

[5] When running Claudien, which is a kind of first order data mining system, many rules are generated. These are usually filtered before being applied for prediction purposes, cf. [5, 2].

[6] These rules were not used in the prediction process due to the doubtfull quality of the induced rules.

Notice that throughout, as different aspects are being learned, and as one predicate may use the other predicates, this is a multiple predicate learning task. Furthermore, many of the rules induced are recursive or even mutually recursive. Though we induced rules taking into account only two notes, the use of recursion makes that this induction problem is *not* easily transformable into attribute value form ! Also, further investigations might want to consider longer chains, done by Dovey, who considered chains of up to 4 notes.

It is our intention to make the data, the generated MIDI-file as well as the settings employed in Claudien publicly available by ftp. For more details, please contact the authors.

5 Prediction

To predict the different aspects of each note needed for performance, we employ a kind of forward reasoning procedure. The reason for using forward reasoning is that the induced hypotheses are highly recursive and typically consist of many rules. This makes a classical backward chaining procedure very inefficient (using Prolog as is, termination problems did arise). Secondly, the prediction process employs two meta-rules in order to handle double predictions and missing predictions. Double predictions arise when for a given note and aspect, two (or more) different values are predicted, missing predictions arise when for a given note and aspect no prediction is made. The meta-rules applied in this case represent general musical knowledge.

The prediction process can be summarized as follows. A typical forward reasoning procedure is stepwise applied. In forward reasoning, whenever the body matches the background knowledge, the corresponding head is asserted.

1. apply the rules induced in step 1 for predicting the melody notes, (forward reasoning)
2. repeatedly apply the rules induced in step 2 for predicting the melody notes (using forward reasoning, i.e. until no new predictions are made)
3. repeatedly apply the rules induced in step 3 for predicting the melody notes (using forward reasoning, i.e. until no new predictions are made)
4. (a) remove double predictions with the continuity rule
   (b) repeatedly apply the default rule to predict missing features of melody notes, and the rules induced in step 2 and 3 to predict missing features of melody notes. (until all features of melody notes are predicted)
5. (a) repeatedly apply the rules induced in step 4 for predicting the non-melody notes
   (b) apply the first default rule for non-melody notes
   (c) apply the second default rule for non-melody notes
   (d) apply the continuity rule for non-melody notes

The meta-rules are described below :

default rule for melody notes: if there exists a prediction for a note but not its successor, then apply the existing prediction also to the successor.

continuity rule for melody notes: if there exists a double (resp. no) prediction for a note, then select that prediction that is closest to the average of its predecessor(s) and successor(s).

first default rule for non-melody notes: if there exists a non-melody note that is related to the same melody-note (in the same way) as a non-melody note for which there is a missing prediciton, then predict the same values for both non-melody notes.

second default rule for non-melody notes: if all of the non-melody notes that are related to a melody note in a certain way have a missing feature, then select the value of a non-melody note that is related to the previous melody-note in the same way.

All of these rules are based on a kind of continuity principle which states that the music typically evolves in a continuous manner. The current meta-rules have been kept as simple as possible.

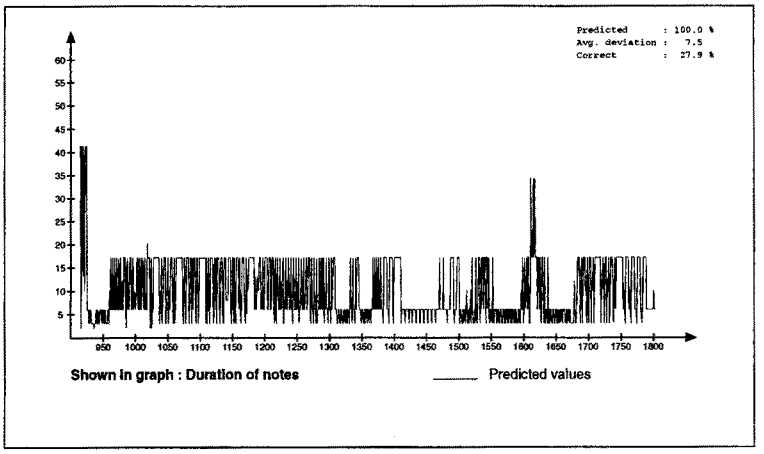
6 Experimental results

The above prediction procedure was applied to the two Ampico rolls (melody notes only), and to the MIDI file (all notes) generated by Carl Verbraeken. For reasons of space, we only present the latter results here (for more details about the Ampico rolls, cf. [11]). Before discussing these results, we should point at a fundamental difficulty in evaluating our results. To this aim, we quote Gerard Widmer [12], when describing a similar evaluation.

> It is difficult to analyze the results in a quantitative way. One should compare the system's performance of a piece with a human performance of the same piece and measure the average differences between the two curves. However, the results would be rather meaningless. For one thing, there is no single correct way of playing a piece. And second, relative errors or deviations cannot simple be added: some notes and structures are more important than others, and thus errors are more or less grave.

There are thus at least two different ways to evaluate our results. First, there is the classical machine learning perspective, which prescribes that individual predictions should be compared with actual values, and their accuracy measured. Secondly, and as Widmer argues, more importantly, one should also evaluate the way the music sounds.

Let us first look at the machine learning perspective. The results of the prediction process are shown in Figures 1, 2 and 3, each focussing on one feature needed for MIDI, i.e. duration of notes, duration of gaps, and force. In these figures, the x-axis represents the sequence of notes played, and the y-axis the actual values, respectively the predicted values.



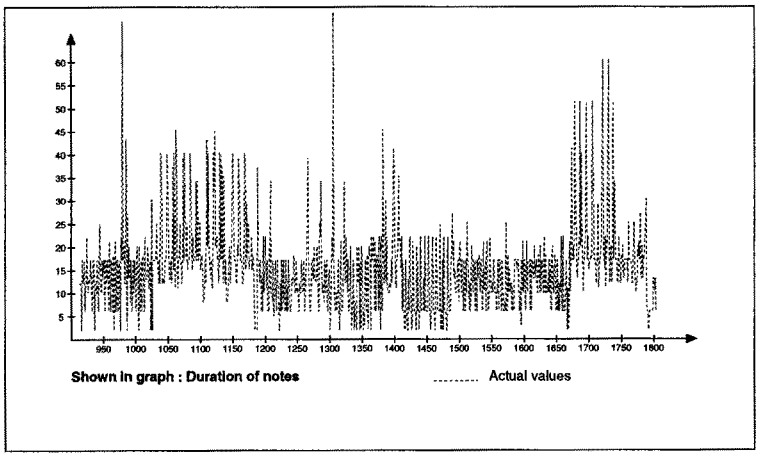
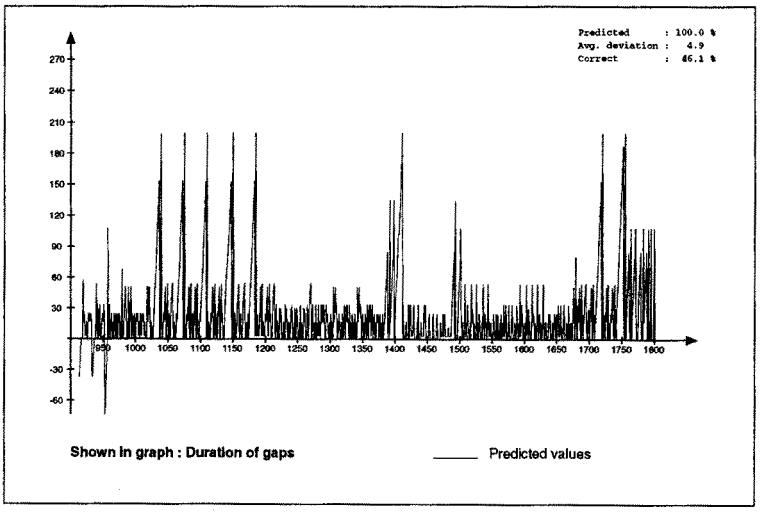


Fig. 1. Duration


Predicted : 100.0 %
Avg. deviation : 4.9
Correct : 46.1 %
270
240
210
180
150
120
90
60
30
-30
-60
950 1000 1050 1100 1150 1200 1250 1300 1350 1400 1450 1500 1550 1600 1650 1700 1750 1800
Shown in graph : Duration of gaps
Predicted values

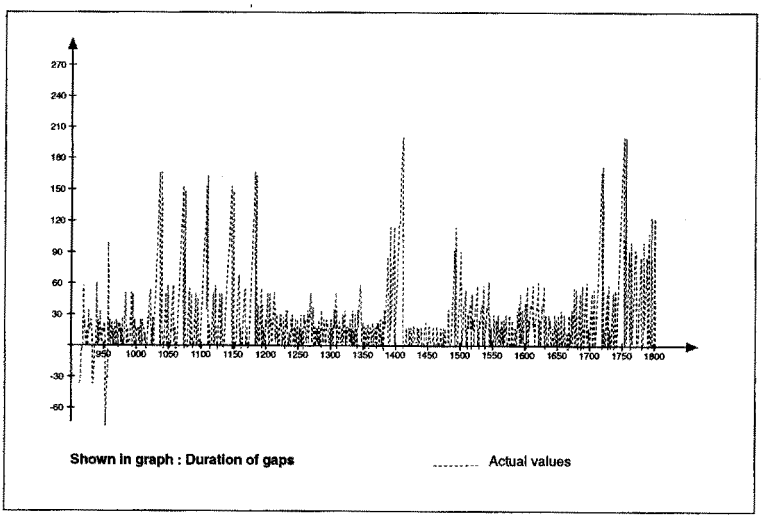
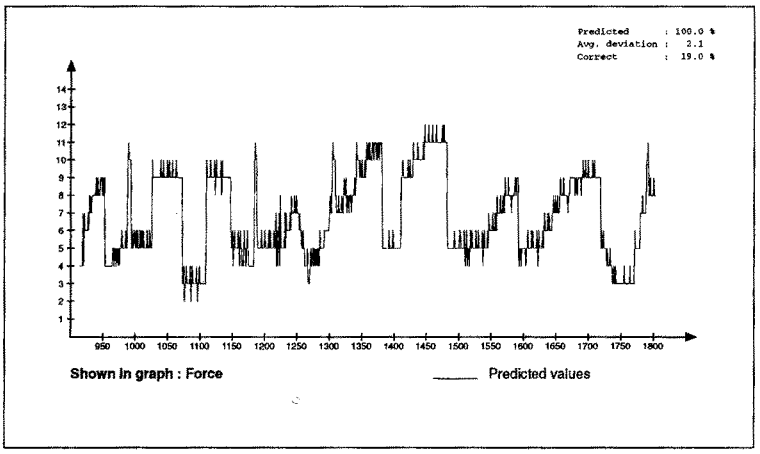

270
240
210
180
150
120
90
60
30
-30
-60
950 1000 1050 1100 1150 1200 1250 1300 1350 1400 1450 1500 1550 1600 1650 1700 1750 1800
Shown in graph : Duration of gaps
Actual values

Fig. 2. Gaps


Predicted : 100.0 %
Avg. deviation : 2.1
Correct : 19.0 %
14
13
12
11
10
9
8
7
6
5
4
3
2
1
950 1000 1050 1100 1150 1200 1250 1300 1350 1400 1450 1500 1550 1600 1650 1700 1750 1800
Shown in graph : Force
Predicted values

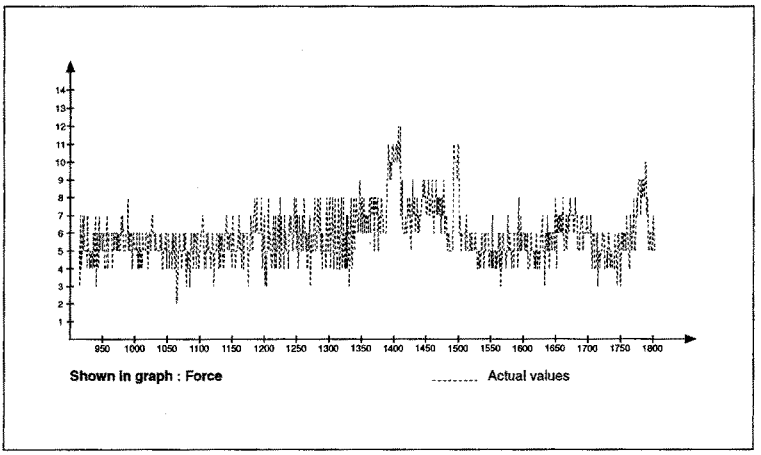

14
13
12
11
10
9
8
7
6
5
4
3
2
1
950 1000 1050 1100 1150 1200 1250 1300 1350 1400 1450 1500 1550 1600 1650 1700 1750 1800
Shown in graph : Force
Actual values

Fig. 3. Force

First, duration of the notes is predicted rather well, the deviation being on the average only 7.5 on a scale of 120 per beat, which can hardly be observed by the human ear. However, important differences arise when looking at the peaks, which are predicted rather poorly. An explanation for this could be that there are too few peaks in the training data and hence that the peaks are considered to be noise by the clausal discovery engine.

Second, duration of the gaps is quite well predicted, the average deviation being smaller than for duration of the notes.

Thirdly, the force. Again the trends are predicted rather well, and the peaks are much harder. However, for force the average deviation is relatively larger than for duration, which indicates that force is much harder to predict, a finding confirmed also by Dovey's experiments. For force, it also turns out that especially the second default rule for non-melody notes contributed many predictions (about 40 per cent of the notes). The other meta-rules were less used.

Fourth, when looking at the individual rules obtained (like Dovey did), it is easily seen that they make sense though they are sometimes quite specific. This is probably due to the use of a single musical piece.

Though the above three figures may be significant from the machine learning point of view, the real issue is how well the predicted performance sounds. To this aim, the predicted features for all the notes were encoded on a MIDI file, so that the human ear could judge. When listening to the MIDI file (which we intend to make available by ftp), there are some interesting observations to make. First, at the global level, the results are rather disappointing. We believe the generated performance is similar to a (not so good) student that just started to practice playing Mendelssohn's Lied Ohne Worte. However, at the level of small parts (a few bars), the sound is sometimes quite good. Secondly, the listener gets the impression that the force is predicted very well but there is often something wrong with the rhythm. This is interesting as it contradicts the machine learning perspective, where according to the curves, force was predicted much worse than duration. Probably the human ear is more sensitive to rhythm than to force. One possible cause for this is the default rules, which take into account only the previous note. In practice, it would be better to also use the next note and compute some kind of average (as with the continuity rule), which we are currently investigating.

When combining the machine learning results with the musical ones, it is easily observed that we are facing a very hard and challenging task. This is so because the learning task involves multiple predicate learning as well as sequence prediction. Indeed, consider a bar of a piece. In order to make the bar sound good, we need to predict the different predicates of each individual note in the bar (sequence) well. This means that the quality of the predictions of a bar crucially depends on many different predictions. Whenever one of these predictions is predicted very poorly, the whole bar (and maybe even more than a bar) will be affected and will sound erroneous.

In sum, we believe that the task addressed in this paper is an extremely challenging problem for inductive logic programming. It is also important, in

that it demonstrates the need for multiple predicate learning and sequence prediction. These are two problems, which are hard to solve with propositional learning techniques, but which have received some atttention within inductive logic programming. Hopefully, our study will motivate further research in this direction.

Another conclusion, we could draw is based on comparing the MIDI results with those of the Ampico rolls, the first produced by Carl Verbraeken, and the second by Rachmaninoff. The results for Ampico are sligthly worse than those for MIDI. A plausible explanation for this is that Rachmaninoff's playing is less predictable than that of Carl Verbraeken, because Rachmaninoff is the better piano player.

7 Conclusions and Further Work

Starting from an initial study by Matthew Dovey aimed at analysing Rachmaninoff's Piano Performances using inductive logic programming techniques, we have obtained several new results. Most importantly, rather than deriving rules for the qualitative interpretation (by a human), we have shown how to induce rules that can be used to generate music performances on MIDI. This required the use of a significantly enriched and modified representation of the performance and structural data used to train the induction engine. Secondly, we have also shown that this learning task (including the prediction) is challenging and motivating for inductive logic programming as it requires multiple predicate learning, sequence prediction, and potentially also number handling. Thirdly, we have demonstrated the use of the clausal discovery engine Claudien.

There are several directions for further research. First, our results could definitely be improved by employing more musical background knowledge. As already indicated by Dovey, models of the human listener, such as those advocated by Narmour or Lerdahl [10, 8] could be useful. Second, the results could also be improved by employing more powerful inductive logic programming techniques. This would include more advanced multiple predicate learners and sequence prediction algorithms, as argued above. Furthermore, number handling techniques would allow to predict the actual values of the features instead of their range. Finally, it might be better - with the current encoding - to use induction algorithms aimed at classification, such as ICL [3], instead of the clausal discovery engine employed.

Acknowledgements

The authors would like to thank Gerard Widmer for pointing us to the work of Matthew Dovey and comments on this work, Matthew Dovey for kindly supplying us with his data and information, Carl Verbraeken for his interest and musical help, and to the members of the Leuven ILP team, in particular to Hilde Ade and Wim Van Laer. This work is also part of the ESPRIT IV project no.

20237 on Inductive Logic Programming II, funded by the European Union. Luc De Raedt is supported by the Belgian National Fund for Scientific Research.

References

1. D. Cope. Experiments in Music Intelligence, In *Proceedings of the International Computer Music Conference*, Computer Music Association, 1987.
2. L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1058–1063. Morgan Kaufmann, 1993.
3. L. De Raedt and W. Van Laer. Inductive constraint logic. In *Proceedings of the 5th Workshop on Algorithmic Learning Theory*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1995.
4. L. De Raedt, N. Lavrač, and S. Džeroski. Multiple predicate learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1037–1042. Morgan Kaufmann, 1993.
5. L. De Raedt and L. Dehaspe. Clausal Discovery, Technical Report, Dept. of Computer Science K.U.Leuven, 1995, Submitted to *Machine Learning*.
6. M.J. Dovey. *Analysis of Rachmaninoff's Piano Performances using Inductive Logic Programming*, Oxford University Computing Lab, 1995. An extended abstract was published in *Proceedings of the 8th European Conference on Machine Learning*, Lecture Notes in Artificial Intelligence, Springer Verlag, 1995.
7. J. LaRue, *Guidelines for Style Analysis*, W.W.Norton and Co., 1970.
8. F. Lerdahl and R. Jackendorff. *A generative theory of tonal music*. MIT Press, 1983.
9. S. Muggleton. Inverse entailment and progol. *New Generation Computing*, 13, 1995.
10. E. Narmour. *Beyond Schenkerism: the need for alternatives in music analysis*. Chicago University Press, 1983.
11. E. Van Baelen. Analyse van piano-uitvoeringen met behulp van inductie (in Dutch). Master's Thesis, Dept. of Computer Science, K.U.Leuven, 1996.
12. G. Widmer. The synergy of music theory and artificial intelligence: learning multilevel expressive interpretation. In *Proceedings of the 11th AAAI*, 1994.