RNBERT: FINE-TUNING A MASKED LANGUAGE MODEL FOR ROMAN NUMERAL ANALYSIS

Malcolm Sailor
Yale University
malcolm.sailor@gmail.com

ABSTRACT

Music is plentiful, but labeled data for music theory tasks like roman numeral analysis is scarce. Self-supervised pretraining is therefore a promising avenue for improving performance on these tasks, especially because, in learning a task like predicting masked notes, a model may acquire latent representations of music theory concepts like keys and chords. However, existing models for roman numeral analysis have not used pretraining, instead training from scratch on labeled data, while conversely, pretrained models for music understanding have generally been applied to sequence-level tasks requiring little explicit music theory, such as composer classification. In contrast, this paper applies pretraining methods to a music theory task by fine-tuning a masked language model, MusicBERT, for roman numeral analysis. We apply token classification to get a chord label for each note and then aggregate the predictions of simultaneous notes to achieve a single label at each time step. The resulting model substantially outperforms previous roman numeral analysis models. Our approach can readily be extended to other note- and/or chord-level music theory tasks (e.g., nonharmonic tone analysis, melody harmonization).

1. INTRODUCTION

Roman numeral analysis is the task of identifying the chords in a piece of music and then indicating their role with respect to the current key. Although it was developed in the European classical tradition, it is an essential element of the musical toolkit for musicians working in many different Western-derived styles (for instance, a jazz musician might speak of a "ii-V to vi ('two-five to six')" or a pop musician might say "the bridge starts on IV").

A small but growing literature has employed deep learning models for automatic Roman numeral analysis of symbolic music (Section 2), training the models from scratch on labeled data. But labeled data for Roman numeral analysis is scarce, whereas symbolic music is comparatively plentiful. Self-supervised pretraining therefore seems likely to yield dividends, especially considering that, in order to perform a self-supervised task, the model can be expected to learn latent representations of music theory concepts like chords and scales: if you need to predict a given musical note, you will do a lot better if you can estimate the expected key and chord. Moreover, as a general matter, it seems likely that it will prove more efficient and more practical for MIR researchers and music theorists to fine-tune large-scale foundation models for specific analytical tasks, rather than training bespoke models from scratch for every task. These considerations inspire the present work, where we fine-tune a masked language model on Roman numeral analysis, obtaining state-of-the-art classification accuracy. [1]

2. RELATED WORK

Whereas automatic chord recognition using audio signals has an extensive literature [1], this paper contributes to a smaller body of work on Roman numeral analysis of symbolic music, which is both an easier and a harder problem. Easier, because working with symbolic data means the model does not need to devote capacity to identifying the sounding pitches, but harder, because Roman numeral analysis requires not only identifying chords but also describing their harmonic function within their tonal context (e.g., rather than simply labeling a chord as "E major", labeling it as "V6/vi in C major"). Details of Roman numeral analysis are beyond the scope of this paper but are covered in any textbook of classical harmony such as [2, 3].

While earlier work employed various approaches for automated Roman numeral analysis, more recently, deep learning has come to predominate, including recurrent models [4–7], transformers [8, 9], and, most recently, graph-based architectures [10]. As far as we know, the best performance in the existing literature has been obtained by *AugmentedNet* [6,7] and *ChordGNN* [10], and we compare our results below with those reported in [6, 10]. [2]

All of these models for Roman numeral analysis are trained from scratch, not making use of self-supervised pretraining. A hitherto separate area of research is pretraining large models for the understanding of symbolic music. Both MusicBERT [11], the model used in this paper, and MidiBERT-Piano [12] pretrain BERT-like [13]

[1] We release the code to reproduce our results at https://github.com/malcolmsailor/rnbert.

[2] Unfortunately, the results of [7] are reported in a manner that makes them difficult to compare directly with these other papers and with our own results.

encoder-only transformers on a masked language modeling task. [14] adds a GPT-like causal language modeling task as well. So far, these pretrained models have mainly been fine-tuned on sequence-level tasks such as composer, genre, or emotion classification. As far as we know, none of these pretrained models have been applied to Roman numeral analysis, or any other problem involving the prediction of explicit music-theoretical labels.

3. EXPERIMENTAL SETUP

3.1 Corpus

To our knowledge, the corpus used in this study is the largest yet assembled for Roman numeral analysis. The major components of this corpus include the various corpora released by the Digital and Cognitive Musicology Lab [15–17], the TAVERN set of theme and variations by Beethoven and Mozart [18], the set of Beethoven Piano Sonatas first movements introduced in [4], and the various other items included in the When in Rome meta-corpus [19], including analyses of Bach preludes and chorales, and a large number of 19th century lieder, including works of women composers. The total contents of this corpus are enumerated in the first line of Table 1.

Data subset	Scores	Notes	Chords
All	1,404	1,289,888	161,473
AugmentedNet v1	347	701,703	77,570

Table 1. Overall contents of the datasets used for training and evaluation.

For a fair comparison with [6, 10], we also train and evaluate on the subset of our data used in those papers, employing the same training/validation/testing splits. [3] We refer to this subset as "AugmentedNet v1" to distinguish it from the somewhat larger dataset used in [7]. Note that, for scores having two analyses (in particular, the scores of the TAVERN dataset, where each score was analyzed by two separate annotators), AugmentedNet v1 includes both versions (following [6]), whereas in the full corpus, we randomly choose only one of the two versions for inclusion. AugmentedNet v1's note count is substantially increased by the inclusion of these duplicate TAVERN scores, which comprise 106,981 notes.

Unlike some prior work (e.g., [6, 9]), we do not experiment with training and/or evaluating on smaller, more homogenous subsets of our corpus (for example, the Beethoven piano sonatas only). Our goal is to train the best and most general Roman numeral analysis model we can and we expect that such a model will be best obtained and evaluated by using as much data as possible.

[3] 7 scores from AugmentedNet v1 were excluded because of preprocessing errors (for example, because they include time signatures not supported by MusicBERT's OctupleMIDI encoding scheme). These scores exclusively came from the training split and so, if their omission has any effect on RNBert's performance relative to that of the other models trained on the AugmentedNet v1 dataset, it should bias it downwards.

3.2 Data representation

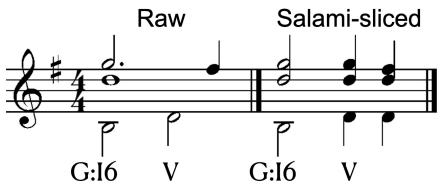

Figure 1. A hypothetical musical example and its analysis, before and after salami-slicing.

Onset	0	0	0	2	2	2	3	3	3
Release	2	2	2	3	3	3	4	4	4
Pitch	59	67	79	62	69	79	62	69	78
RN	I6	I6	I6	V	V	V	V	V	V
Key	G	G	G	G	G	G	G	G	G

Table 2. The example from Figure 1 in tabular format after salami-slicing. Time signatures, tempi, and bar lines are omitted.

The scores associated with the analyses in our dataset are encoded in a variety of formats, such as MusicXML (.xml or .mxl), MuseScore (.mscx), or Humdrum (.krn). We convert these into a tabular format, illustrated in Table 2, labeling each note with the associated key and chord annotation. Note that, because we use a model that was pretrained on MIDI data, which does not specify pitch-spelling, our pitch inputs are not spelled (that is, they use midi numbers like "78" rather than pitch names like "F#5"). For further discussion, see Section 3.3.1.

When reading scores, we apply several preprocessing steps.

First, following MusicBERT, the score is quantized at the 64th note level.

Second, we "salami-slice" the score: at each timestep with one or more onsets or releases, we split any ongoing notes into two, in order to obtain a purely homophonic rhythmic texture in which all onsets and releases are synchronized across all parts. (The term "salami-slicing" is due to [20].) Salami-slicing is necessary to ensure that each note belongs to only one chord. Otherwise, a note may persist through multiple changes of chord, either because it is a common tone among each of the chords (like the alto D in Figure 1), or because it realizes a suspension or similar dissonant idiom (like the soprano G in Figure 1). Fortunately for our purposes, salami-slicing should not affect harmonic analysis, because it does not change the pitch content of the score. Moreover, musical idioms like suspensions and pedal tones that cross changes of chord are analyzed the same whether or not they are tied or sounded anew at the onset of the new chord.

Third, we dedouble the notes of the score, removing any notes that have the same pitch, onset, and release (regardless of whether they are performed by the same instrument). Dedoubling has two advantages. First, it reduces the sequence length. Second, it produces a more homogeneous texture between music for small ensembles,

where pitch doubling is less common, and music for large ensembles like orchestras, where pitch doubling is ubiquitous. Such homogeneity is particular desirable since nearly all the available labeled scores are small-ensemble works like piano sonatas and string quartets, but we would like our model to generalize to large-ensemble works like symphonies and operas.

MusicBERT has a maximum sequence length of 1000 tokens. Therefore, in both training and evaluation, we crop scores into segments of 1000 tokens, stepping through the score with a hop size of 250.

3.3 Task

Roman numeral analysis involves labeling chords with integers (Roman numerals) indicating their root with respect to the scale of the current key (e.g., "V" in C major indicates that the root is G, the 5th scale degree). Figured-bass numerals are typically appended to to indicate the chord's inversion (e.g., "6" for first inversion). While the quality of the chord is often assumed to conform to the scale, alterations of quality can be indicated according to a variety of conventions (e.g., upper-/lower-case for major/minor, appending "+" or "o" to indicate augmented or diminished chords, or combining figured-bass numerals with accidental signs). The Roman numerals can be prefixed with accidentals to indicate altered scale degrees (for example, bVI in C major would be A-flat). "Tonicizations"—that is, one or more chords borrowed from another key—can be indicated by "secondary" Roman numerals, typically following a slash, as in "V/ii", which in C major would indicate the V chord of d minor. Finally, since the Roman numeral only indicates a harmony with respect to some key, for a Roman numeral to be meaningful, we need to indicate the key as well.

Since a complete Roman numeral consists of multiple distinct elements, the combinatorial space of these elements is very large. Encoding each distinct combination as a token, would require a large and sparse vocabulary, posing challenges for training and generalization. Therefore, the approach adopted here and elsewhere is to treat Roman numeral analysis as a multitask learning problem, where we predict the key, quality, inversion, and degree separately. (The degree is sometimes further decomposed into "primary" and "secondary" components, but in the current work, we predict these jointly.) It should be noted, however, that this multitask may approach obtain a smaller vocabulary size at the expense of some coherence among the different elements of the Roman numeral. For example, suppose there is a passage that is ambiguous between I and vi6 (two chords which share two of their three pitch-classes as well as the same bass note). If the model distributes the probability roughly equally between the two possibilities it may easily occur that the inversion and degree predictions "decohere" and we end up with a plainly incorrect prediction like I6 (rather than I) or vi (rather than vi6). (One solution to this problem of decoherence was proposed by [21], which we discuss in Section 3.6 below.)

3.3.1 Pitch spelling

One difference between our approach and some prior work (e.g., [5, 7]) is that MusicBERT uses unspelled pitch inputs (midi numbers like "67") rather than letter names (like "F#5"). Our output key predictions are therefore also unspelled (e.g., pitch-class 6, rather than "F-sharp") because, with unspelled inputs, the output spelling is undefined. [4]

We consider our model's inability to predict spelled keys unimportant. Given spelled inputs (e.g., pitches like "Db5" rather than MIDI numbers like "61") and an unspelled key (e.g., "1 major"), predicting a spelled key (e.g., "Db major") is trivial and could likely be performed with perfect accuracy by a rule-based algorithm (e.g., taking the enharmonically equivalent key closest to the centroid of the spelled pitches on the "line of fifths" [23]). Moreover, keys with plausible enharmonic equivalents (like F-sharp major or E-flat minor) are rarely used. Their classification is therefore unlikely to significantly affect validation/test performance.

3.4 Data augmentation

We employ two data augmentation techniques on the training data. First, we transpose each score to all 12 keys of the chromatic scale. Second, we create a version of each score with the durations scaled by a factor of 2. If the mean duration in the score is greater than the mean duration of the training set as a whole, we scale its durations down by 2; if it is less, we scale them up by 2.

We experimented with adding synthetic data similar to the procedure introduced in [6, 7] and also adopted in [10]. However, we did not find that it improved the model performance. It is possible that synthetic data was less helpful in our case than with AugmentedNet [7] because, whereas for that model, the inputs consist of pitch-vectors at each time step, for our model, the inputs are simply the notes of the score, and therefore the difference between synthetic and real data is more apparent to the model.

3.5 Model

3.5.1 MusicBERT

The model that we fine-tune, MusicBERT [11], is a bidirectional transformer encoder pretrained on a masked language modeling task. The dataset for pretraining is a corpus of over 1 million midi files, 3 orders of magnitude larger than our Roman numeral dataset. This difference in scale motivates the use of a pretrained model. We chose MusicBERT for our experiments because, of the pretrained symbolic music models of which we are aware, it used the largest pretraining dataset (by comparison, [12] pretrains on fewer than 5,000 scores of exclusively piano music) and also because of the elegance of the OctupleMIDI scheme MusicBERT uses to encode its inputs. A more detailed

[4] This is because the only difference between enharmonically equivalent keys like F-sharp and G-flat is how they are notated. Certain pieces of music, such as Fugue no. 8 of Bach's Well-tempered Clavier, Book 1, have even been variously printed in two enharmonically equivalent keys [22].

comparison fine-tuning symbolic music models for Roman numeral analysis will have to await further work.

In the OctupleMIDI encoding, eight features of a musical note are first embedded individually: time signature, tempo, bar number, metric position within the bar, pitch, duration, and velocity. [5] To obtain a single input at each time step, these eight embeddings are concatenated and then projected to the model's embedding dimension. OctupleMIDI reduces the sequence length when compared with other encoding schemes like Midi-like [24,25], REMI [26], or Compound Word [27]. This reduction occurs because in OctupleMIDI, tokens and notes are in one-to-one correspondence, whereas the other schemes use tokens for other items such as time-signatures or barlines. For our use case, the fact that all tokens correspond to notes has the added virtue that we do not waste computations classifying non-note tokens.

In our experiments, we use the MusicBERT "base" architecture, whose hyperparameters are modeled on those of the BERT base architecture ([13]), with a hidden dimension of 768, 12 layers, and 12 attention heads. We use the pretrained checkpoint provided by [11]. For further details we refer the reader to the original MusicBERT paper.

MusicBERT is not trained solely or even mainly on Classical music, whereas our annotated data consists entirely of Classical music. This does not seem likely to be a problem, because in the first place, a great deal of the tonal idiom (i.e., keys, chords) is shared between different styles of tonal music, and tonal music surely predominates in MusicBERT's training set. Moreover, to pretrain on only classical music would mean greatly reducing the amount of training data, as it's extremely unlikely that 1,000,000 distinct midi files of Classical music exist. ([20] is based on what is to our knowledge the largest corpus of Classical midi files in existence and features under 15,000 files.)

3.5.2 Token classification

To perform Roman numeral analysis, we adopt a token classification approach, predicting the key and Roman numeral for each token in the input. Since each token corresponds to a note, this amounts to predicting the chord during which each note occurs. While training, we calculate the loss on a per-token basis. In evaluation, in order to obtain a single prediction for each salami slice, we average the logits of simultaneous notes.

The token-classification heads are two-layer multilayer perceptrons (MLPs) whose inner dimension is the embedding dimension of the model (768, in our experiments).

To obtain the overall loss, we simply take the mean of the cross-entropy loss for each individual task. We tried learning a weighting of the contribution of each task to the global loss, following the approach introduced by [28] and implemented in an MIR context by [29], but observed a small degradation in model quality when doing so.

[5] Midi velocity is encoded in the OctupleMIDI format but is absent from the symbolic scores of our dataset. Therefore, we use a default value of 96 for all note velocities in our dataset.

3.5.3 Fine-tuning procedure

In our fine-tuning experiments, we found it important to freeze parts of the model to reduce the number of trainable parameters and avoid overfitting. Freezing the first 9 layers of MusicBERT seemed to give the best results. The parameter counts are given in Table 3.

We used a learning rate of 2.5×10^{-4} with a linear warmup of 2500 steps followed by a linear decay of the learning rate to 0. When training on multi-task Roman numeral classification, we fine-tuned for 50,000 steps. When training on key classification only (see Section 3.6), we fine-tuned for 25,000 steps.

When experimenting with varying these hyperparameters, we did not typically find their precise values to have much effect on the performance of the model. This implies that the fine-tuning is fairly robust to different hyperparameter choices.

Model	Total	Trainable
Base	108,805,598	26,782,729
Key conditioned	111,023,816	28,859,891

Table 3. RNBert parameter counts.

3.6 Key conditioning

In preliminary versions of RNBert, we found that high entropy of the output probabilities for the degree task seemed to mainly occur in two distinct scenarios. The first scenario involved unusual or hard-to-analyze chords, where high entropy is to be expected. The second scenario involved chords that, given a key, were straightforward to analyze, but where the model appeared to be uncertain about the choice of key. [6] To illustrate this latter scenario, suppose we are analyzing a passage, and we recognize an A minor chord, but we are uncertain whether the key is C major or G major. In that case, though we will not know whether to label it with "vi" or "ii," this uncertainty isn't about the chord itself, but only about the key. If the model distributes the probability mass roughly evenly between the two possibilities, it may emit an incoherent composite prediction like "ii of C major" (a D minor chord) or "vi of G major" (an E minor chord). In general, the degree task depends on the key task in this way. Therefore, in some of our experiments, we made the Roman numeral prediction by conditional on the key.

In these key-conditioned experiments, we embed the key tokens with a two-layer MLP with hidden and output dimensions of 256 and GELU activation. [7] We then

[6] We do not have space to discuss this further, but there are music theoretic reasons to think that a certain degree of uncertainty about key annotations is inevitable because the key of certain passages, especially transitional ones, can be analyzed in more than one way.

[7] In principle, we should be able to replace this MLP with a simple embedding layer and obtain the same results. In practice, however, we found that using a simple embedding layer barely improved performance above the unconditioned baseline, even with teacher forcing. We suspect that this occurs because the loss landscape of the MLP has better training dynamics. After training, on the other hand, it should be possible to replace the MLP with an embedding table that simply encodes the output

concatenate this key embedding with the output from MusicBERT to obtain the input to the Roman numeral classification heads.

In training, we employ teacher forcing, that is, we condition on the ground-truth key annotations from the labeled data. In evaluation, we first predict the key with a separately fine-tuned model, then condition the chord predictions on these predicted keys.

Another attempt to encourage coherence between key and Roman numeral predictions is [21], who use a neural autoregressive distribution estimator (NADE). Their approach extends beyond ours insofar as it conditions each sub-task of the Roman numeral classification on the previous tasks. In preliminary experiments applying a similar approach to RNBert, we observed a small decline in performance across all metrics. We defer to future work a qualitative evaluation of these predictions and further similar experiments.

3.7 Post-processing steps

In postprocessing, we collate the predictions from each segment, combining the overlapping logits of adjacent segments by linearly interpolating between them. (By analogy to audio signal processing we could say that we cross-fade between the logits of neighboring segments.) We then average the logits of simultaneous notes to obtain a single set of logits for each salami-slice.

To avoid implausibly brief key changes of one or two salami-slices' duration (which otherwise sometimes occur at transitions between keys, when the model estimates both keys to be approximately equiprobable), we use a dynamic programming approach to decode the key predictions. Specifically, we employ the Viterbi algorithm, using RNBert's output probabilities as the emission probabilities and defining a transition probability matrix that is uniform, except for self-transitions, whose probabilities are upweighted. This decoding scheme has a negligible effect on the measured accuracy of the predictions, while effectively eliminating implausibly brief key changes.

4. RESULTS AND DISCUSSION

Table 4 provides our results, expressed following [6] as the proportion of time the predicted labels are accurate, with 32nd-note resolution. We give two sets of results: on lines 1 to 3, training on the dataset and training/validation/testing splits used by [6, 10] for a fair comparison with these prior papers, and on lines 4 to 6, training on our complete dataset.

Concerning the composite "RN" labels, $\text{RN}_{\text{-root}}$ refers to the conjunction of degree, quality, inversion, and key, while $\text{RN}_{\text{+root}}$ adds to these the chord root. Predicting the root is redundant: the root of a Roman numeral is a deterministic function of the degree and key (e.g., the root of #iv in C major is F-sharp). There is hence no need to include it in the composite Roman numeral, or indeed, to predict it at all. Therefore, when training RNBert on our full dataset, we do not predict the root and report only $\text{RN}_{\text{-root}}$. When training on the AugmentedNet v1 data subset, in contrast, in order to ensure a fair comparison with the prior models, both of which predicted the root, we train RNBert to predict the root and report the results for both $\text{RN}_{\text{+root}}$ and $\text{RN}_{\text{-root}}$. It can be seen that the inclusion or exclusion of the root makes almost no difference, as one would expect. Finally RN_{alt} refers to an alternate task learned in [6, 10], replacing the quality, degree, and root predictions with a vocabulary of the 75 most common Roman numerals in the AugmentedNet v1 training set. We did not train RNBert on this task, but we report the prior results on it to facilitate comparison with our results.

When training on the AugmentedNet v1 subset (Table 4, line 3), RNBert substantially outperforms the prior models on degree and quality. However, it outperforms the earlier models by a much more substantial margin when predicting the composite Roman numeral $\text{RN}_{\text{+root}}$. This implies that there is more coherence among the various dimensions of its predictions. Such coherence may be due to the robustness of the representations MusicBERT learns in its pretraining. It implies that, even where RNBert's predictions don't agree with a human annotator's, they are more likely to be useful, since they are more likely to be internally consistent.

When training on the complete dataset (Table 4, lines 4–6), RNBert exceeds the performance of the earlier models by an even larger margin, especially when predicting the composite Roman numeral. We defer a discussion of the effect of key conditioning to Section 4.1.

One thing to note about these results is that, while the models on lines 4 and 5 greatly exceed the performance of the models on lines 1–3 on degree, quality, inversion, and $\text{RN}_{\text{-root}}$ prediction, when it comes to key prediction, the AugmentedNet v1-trained models actually perform better (with the exception of the ChordGNN model). We believe this occurs because key prediction on the subset is simply an easier problem, since it contains less music from the late 19th century and beyond, a period when music tended to modulate more widely.

4.1 Effect of key conditioning

In Table 4, line 6, it can be seen that conditioning the Roman numeral prediction on the ground-truth key (i.e., teacher forcing) has a large effect on degree accuracy. This constitutes a sanity check that the conditioning works as expected: if the model knows the key of the annotation, its ability to predict the Roman numeral's degree shoots up. By contrast, key conditioning, with or without teacher forcing, has little effect on the "quality" and "inversion" metrics. This is also expected, since these tasks do not depend on the key: a first-inversion minor chord is a first-inversion minor chord regardless of the key in which it occurs.

It is somewhat harder to interpret a comparison of RNBert, conditioned on the predicted key (line 5), with the

of the MLP for each key. However, the MLP contributes such a small proportion of the model's overall parameter count that we did not bother to do so.

	Model	Accuracy						
		Degree	Quality	Inversion	Key	RN_{+root}	RN_{-root}	RN_{alt}
	AugmentedNet v1 data subset							
1	AugmentedNet [6]	.67	.797	.788	**.829**	.464		.515
2	ChordGNN+(Post) [10]	.714	.784	**.803**	.813	.518		.529
3	RNBert (key conditioned)	**.731**	**.819**	.796	.825	**.574**	**.575**	
	All data							
4	RNBert (unconditioned)	.762	.867	.872	.822		.620	
5	RNBert (key conditioned)	.749	.864	.872	.823		.624	
6	RNBert (key conditioned, teacher forcing)	.859	.865	.872	N/A		N/A	

Table 4. Accuracy of RNBert and two prior models. The meanings of RN_{+root}, RN_{-root}, and RN_{alt} are described in Section 4. In the model comparison of lines 1–3, we indicate the best metric in **bold** type. Because the teacher-forcing model on line 6 does not predict key, and RN prediction involves key prediction, we do not report RN results for this model.

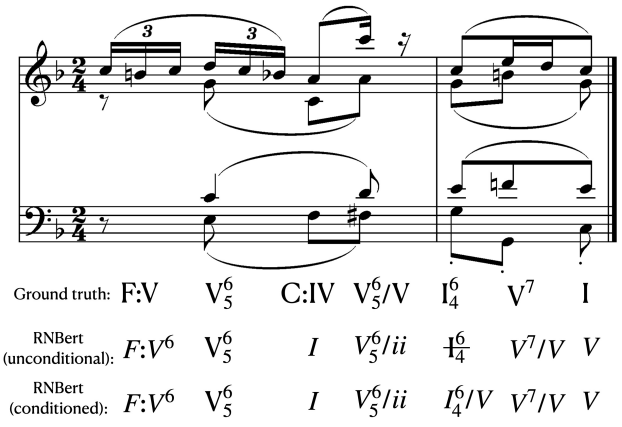

Figure 2. Beethoven, String Quartet in F major, op. 18, no. 1, iv, mm. 7–8. Arguably correct predictions that do not agree with the human annotations are printed in *italic* type and serve to illustrate the discussion in Section 4.2. The prediction printed in ~~strikethrough~~ type is straightforwardly incorrect and illustrates the discussion in Section 4.1.

unconditonal RNBert (line 4). The unconditional model does better predicting degree, but the conditioned model does better predicting the composite RN_{-root}. These results make sense if key conditioning makes the key and Roman numeral predictions more coherent with one another. Even when the unconditional model does not predict the labeled key, it should still predict the labeled degree some proportion of the time, causing its degree accuracy to be higher. The conditional model, on the other hand, should be less likely to do this, but its composite RN prediction should be more coherent and thus more accurate.

Figure 2 can serve as an illustration. The two RNBert analyses are almost identical, being in the key of F major throughout, with tonicized dominant chords at the half cadence that concludes the example. The lone difference occurs at the cadential 64 chord on the downbeat of the second measure. Here, while the conditioned model gives the correct annotation I64/V, the unconditioned analysis gives I64, which is incorrect, since this is a C major chord, and the annotated key is F. The unconditioned model's key and Roman numeral predictions are each plausible on their own—I64 is the most common annotation for a cadential 64 chord—but they do not cohere with one another. And yet, in spite of being incorrect with respect to its predicted key, this I64 prediction happens to agree with the ground truth, and thus the degree accuracy of this example is (spuriously) higher for the unconditioned model.

4.2 The problem of multiple acceptable analyses

One important problem in evaluating Roman numeral analysis models is that there is often more than one correct analysis of a musical passage, so that a model's predictions can be labeled "inaccurate" even when they present valid alternate readings. For example, this may occur with brief passages in another key, which can be analyzed as either modulations (indicated by a change of key) or as tonicizations (indicated with secondary Roman numerals).

On a priori grounds, as well as based on qualitative sampling of the model's predictions, we suggest that a high proportion of RNBert's "inaccurate" predictions are likely to be acceptable alternate analyses. For example, in Figure 2, it is reasonable to analyze the half cadence that concludes the example as a brief modulation to C, as the human annotator did, or as a tonicization, as done by both versions of RNBert. Either analysis is acceptable, but the divergence means that nearly all labels in RNBert's analysis do not agree with the ground truth, in spite of being arguably correct. These considerations may place a ceiling on the accuracy of all Roman numeral analysis models.

5. CONCLUSION

At the broadest level, our results imply that, by fine-tuning pretrained models, we can obtain state-of-the-art performance on music theory tasks. In the specific case of Roman numeral analysis, we suggest that Roman numeral analysis models have now matured to the point where they are ready to be used in large-scale musicological studies. Finally, we note that the approach described in this paper could be readily extended to many other music theory tasks that can be conceived of as a labeling of the notes of a score, including the analysis of dissonant idioms (suspensions, passing tones, and the like) or melody harmonization (that is, labeling each pitch of a melody with a chord).

6. REFERENCES

[1] J. Pauwels, K. O'Hanlon, E. Gomez, and M. B. Sandler, "20 Years of Automatic Chord Recognition from Audio," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2019.

[2] E. Aldwell, C. Schachter, and A. C. Cadwallader, *Harmony & Voice Leading*, 4th ed. Schirmer/Cengage Learning, 2011.

[3] S. M. Kostka and D. Payne, *Tonal Harmony, with an Introduction to Twentieth-Century Music*, 5th ed. McGraw-Hill, 2004.

[4] T.-P. Chen and L. Su, "Functional harmony recognition of symbolic music data with multi-task recurrent neural networks." in *Proceedings of the International Society for Music Information Retrieval Conference*, 2018, pp. 90–97.

[5] G. Micchi, M. Gotham, and M. Giraud, "Not All Roads Lead to Rome: Pitch Representation and Model Architecture for Automatic Harmonic Analysis," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 42–54, 2020.

[6] N. N. López, M. Gotham, and I. Fujinaga, "AugmentedNet: A Roman Numeral Analysis Network with Synthetic Training Examples and Additional Tonal Tasks." in *Proceedings of the International Society for Music Information Retrieval Conference*, 2021, pp. 404–411.

[7] N. N. López, "Automatic Roman Numeral Analysis in Symbolic Music Representations," Ph.D. dissertation, McGill University (Canada), 2022.

[8] T.-P. Chen and L. Su, "Harmony Transformer: Incorporating chord segmentation into harmony recognition," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2019.

[9] ——, "Attend to Chords: Improving Harmonic Analysis of Symbolic Music Using Transformer-Based Models," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 1–13, 2021.

[10] E. Karystinaios and G. Widmer, "Roman Numeral Analysis with Graph Neural Networks: Onset-wise Predictions from Note-wise Features," in *Proceedings of the International Society for Music Information Retrieval Conference*. arXiv, 2023.

[11] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu. (2021) MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training. [Online]. Available: http://arxiv.org/abs/2106.05630

[12] Y.-H. Chou, I.-C. Chen, C.-J. Chang, J. Ching, and Y.-H. Yang. (2021) MidiBERT-Piano: Large-scale Pre-training for Symbolic Music Understanding. [Online]. Available: http://arxiv.org/abs/2107.05223

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2019. [Online]. Available: http://arxiv.org/abs/1810.04805

[14] Z. Li, R. Gong, Y. Chen, and K. Su, "Fine-Grained Position Helps Memorizing More, a Novel Music Compound Transformer Model with Feature Interaction Fusion," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, pp. 5203–5212, 2023.

[15] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, "The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets," *Frontiers in Digital Humanities*, vol. 5, p. 16, 2018.

[16] J. Hentschel, M. Neuwirth, and M. Rohrmeier, "The Annotated Mozart Sonatas: Score, Harmony, and Cadence," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 67–80, 2021.

[17] J. Hentschel, Y. Rammos, M. Neuwirth, and M. Rohrmeier, "An Annotated Corpus of Tonal Piano Music from the Long 19th Century," 2022. [Online]. Available: https://zenodo.org/records/10171721

[18] J. Devaney, C. Arthur, N. Condit-Schultz, and K. Nisula, "Theme And Variation Encodings with Roman Numerals (TAVERN): A New Data Set for Symbolic Music Analysis." in *Proceedings of the International Society for Music Information Retrieval Conference*, 2015, pp. 728–734.

[19] M. Gotham, G. Micchi, N. N. López, and M. Sailor, "When in Rome: A Meta-corpus of Functional Harmony," *Transactions of the International Society for Music Information Retrieval*, vol. 6, no. 1, pp. 150–166, 2023.

[20] C. W. White and I. Quinn, "The Yale-Classical Archives Corpus," *Empirical Musicology Review*, vol. 11, no. 1, pp. 50–58, 2016.

[21] G. Micchi, K. Kosta, G. Medeot, and P. Chanquion, "A deep learning method for enforcing coherence in Automatic Chord Recognition." in *Proceedings of the International Society for Music Information Retrieval Conference*, 2021, pp. 443–451.

[22] P. Badura-Skoda, *Interpreting Bach at the Keyboard*. Oxford University Press, 1995.

[23] D. Temperley, "The Line of Fifths," *Music Analysis*, vol. 19, no. 3, pp. 289–319, 2000.

[24] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer," 2018. [Online]. Available: http://arxiv.org/abs/1809.04281

[25] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, vol. 32, no. 4, pp. 955–967, 2020.

[26] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020.

[27] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[28] L. Liebel and M. Körner. (2018) Auxiliary Tasks in Multi-task Learning. [Online]. Available: http://arxiv.org/abs/1805.06334

[29] J. Qiu, C. L. P. Chen, and T. Zhang. (2022) A Novel Multi-Task Learning Method for Symbolic Music Emotion Recognition. [Online]. Available: http://arxiv.org/abs/2201.05782