Improving chord prediction in Jazz music using note information

Nerea Carbonell, Maximilian Gangloff, Daniel Morales
CS-433 Machine Learning, EPFL, Switzerland

***Abstract*—This paper studies the prediction of chords in Jazz music using long short-term memory networks (LSTM). The goal is to find out whether adding melody information improves sequential chord prediction. Hence, this study compares performances among two models: a baseline only using chord sequences, and a model that uses information on both previous chords and melody notes played. The data used for this study is from the Weimar Jazz Database (WJazzD) [1]. Besides achieving a chord prediction accuracy of 53.89% when including melody information, a 3% improvement over the baseline, we study how melody information affects the sequential chord prediction. Code available at `github.com/nereaiscamu/ML_project2`.**

I. Introduction

A melody is a linear sequence of notes that are perceived as a single entity. It is very common in Jazz that musicians improvise the melody on the basis of chord progressions. Thus, the presumption can be made that the melody is correlated to the chords played and that it aids for the prediction of the next chord.

Over the years, different machine learning algorithms have been used for music composition and real time improvisation [2]. RNNs and LSTM have been explored, with the latter model being most used as it can capture long-term structure of music and yield more coherent results [3].

In this study, we train different LSTM models using either only chords (baseline model) or adding melody and bass pitch information to the chord sequence. Then, we analyze the effects of the melody on the chord prediction for specific sequences to investigate in which cases does the melody help. We used Pandas for the data processing and PyTorch to deploy and train our models.

II. Data

The data used in this study originates from the publicly available Weimar Jazz Database (WJazzD) [1]. We are using two tables from this database. The *beats* table, which corresponds to the annotation of the chords and the bass pitch, and the *melody* table, which is the main table that contains all the notes played for each beat and their duration. In total, we have 456 Jazz songs which have up to 377 chords and between 43 and 4954 notes. We have a total of 418 different chords, many rarely used. Repeated chords are combined in pre-processing. For our chord prediction task we reduce the chord vocabulary, in a trade-off between fidelity to original chord and feasibility of a prediction model.

III. Methods

A. Chord encoding

Chords are sets of notes played together on an instrument. Chords are categorical data, hence before using the information as input for our model, they shall be converted into numerical vectors. Looking at the chord symbols from the *beats* table, we can differentiate the root note, root type and bass (e.g. in $G\sharp 7/D\sharp$ the root note is $G\sharp$, the root type 7 corresponding to a dominant 7, and the bass is $D\sharp$). For the chord encoding, we decided to exclude the bass information. Then, we used a multi-hot encoding approach using the three following one-hot vectors:

1) Root pitch
2) Triad form
3) Chord extension

In the encoding of root pitch to a one-hot vector, to avoid having repeated notes written differently (e.g., $A\sharp$ and $B\flat$), we mapped all flattened pitches to their en-harmonic equivalent ones. The resulting possible root pitches are C, $C\sharp$, D, $D\sharp$, E, F, $F\sharp$, G, $G\sharp$, A, $A\sharp$, B.

In the additional chord information, the 9^{th}, 11^{th} and 13^{th} notes played in the chords (sharpened and flattened pitches included) were mapped to their corresponding 7th or 6th chord. For instance, $79\flat$, $13\sharp$, $913\flat$ are mapped to 7 and 69 is mapped to 6. This chord pre-processing lead to a total number of 156 mapped chords, which will be the number of output classes.The complete chord dictionary can be found at Appendix B.

The triad vector was formed by mapping each chord to its corresponding triad: diminished, half-diminished ($7\flat5$ chords), minor, suspended, augmented and major.

Finally, the extension vector is composed by minor 7^{th}, diminished 7^{th}, major 7^{th}, 6^{th} or "none".

The resulting multi-hot encoding is formed by three vectors of length 12, 6 and 5 respectively.

B. Melody and bass encoding

Combining *beats* and *melody* tables, we obtained all melody notes and duration corresponding to each chord. The melody encoding only represents the pitch class (not the octave) and goes from 0 to 11, where 0 is a C and 11 is a B. This also applies to the bass note encoding.

The melody vector is a sum of the one-hot vectors of each note played during a chord. We also explored to weight the notes by their position in the sequence or by their duration, the listing of models tried is available in III-D. In all variants, the vector is normalized.

In the case where no melody was available, the melody was encoded by a vector of only $0s$ thus having no affect on the sum of the vectors. After the creation of the melody embedding, it is concatenated with the multi-hot encoded chord to generate the input for the model. The same procedure was used for the bass pitch.

C. Model architecture

For the present task, the model is required to model correlations between elements in series. We choose to use a directional LSTM [4] for its ability to model both short and long term dependencies. The model receives sequentially each sample in the series as input, and predicts the next one.

The input to the model is a multi-hot encoded chord as described in III-A, and optionally a vector encoding melody and/or bass notes played during the chord, as described in III-B. We define our baseline LSTM to be of $d = 192$ hidden dimensions and $l = 2$ layers. The input is embedded into d dimensions using a linear layer followed by a ReLU and then fed into the LSTM. A final linear layer maps the LSTM output to a vector the size of the chord vocabulary. The chord prediction corresponds to the highest value in this vector.

D. Models

We want to test whether adding melody and/or bass information to the input of the model helps in the chord prediction task. For that, we compare the following models:

- **Baseline.** Only chord information.
- **Melody.** Chord + a vector which represents the number of times each note was played during the chord, i.e., a bag of notes.
- **Melody weighted.** Same as *Melody*, but weighting notes by $\{2^0, 2^1, ..., 2^9\}$ to give more importance to last notes played.
- **Melody duration.** Notes are weighted both by the geometric sequence and the duration of the note played.
- **Bass.** Chord + a vector that represents the notes played by the bass.
- **Melody + Bass.** Chord + both melody and bass notes played during the chord.

IV. Experiments

A. Training

We split data in train, validation and test sets in $80/10/10$, resulting in 345, 43 and 44 songs respectively. For a fair comparison, the same random split is used to train and evaluate all models. Validation loss is used to perform early stopping with a patience of 15 epochs, otherwise training finishes at 200 epochs. The optimizer of choice is *Adam* [5]. If no regularization is applied, we identify a large generalization gap by comparing the train and validation losses. In addition to early stopping, we use weight decay and dropout as regularizers. Despite regularization, the generalization gap was large, obtaining training accuracy of around 90% accross all models.

B. *Hyperparameter tuning*

For every model, we perform a two-step tuning of hyperparameters. Firstly, we find the best architecture by grid search over the hidden dimensions d and number of layers l. We try values of $d = \{64, 96, 128, 192\}$ and $l = \{1, 2, 3, 4\}$. For the grid search, we fix the learning rate to $lr = 0.01$, weight decay to $wd = 10^{-5}$ and dropout probability to $p = 0.2$. We find the optimal architecture to be $d = 192$ and $l = 2$ in all models. Afterwards, with the optimal architecture, we perform a random search for the training hyperparameters by selecting random values in the intervals $lr = [0.0001, 0.03]$, $wd = [10^{-7}, 10^{-3}]$ and dropout with $p = [0, 0.5]$. Furthermore, we decay the learning rate by half every 50 training epochs. We select the model with highest validation accuracy and report results on the unseen test set. Due to computational constraints, we do not use k-fold cross validation in this search. In Table I we present the optimal values found.

Model	d	l	lr	wd	p
Baseline	192	2	0.007	5e-6	0.2
Melody	192	2	0.017	8e-5	0.2
Melody weighted	192	2	0.002	1e-5	0.1
Melody duration	192	2	0.001	1e-5	0.2
Bass	192	2	0.007	5e-6	0.26
Melody + Bass	192	2	0.004	3e-5	0.31

Table I: Optimal hyperparameters

C. *Results and Discussion*

Chord accuracy. Table II shows the test accuracy obtained for the different models. We observe how all the models using melody and/or bass improve with respect to the baseline accuracy of 50.89%. The best among the proposed models is *Melody*, which improves the baseline accuracy by about 3%. Interestingly, the *Melody + Bass* does not perform as well, indicating that bass information may be somewhat redundant with the melody. However, these results are for a fairly small test set of 44 songs, a total of 2405 chords. We have run 10 models with different random seeds and obtained a standard deviation in the mean accuracy of ± 3.61 and ± 3.71 for baseline and melody models respectively. This variance, given the accuracy difference between models, indicated that more thorough experiments should be conducted to extract meaningful conclusions.

Model	Test accuracy (%)
Baseline	50.89
Melody	**53.93**
Melody weighted	53.47
Melody duration	53.26
Bass	52.72
Melody + Bass	53.68

Table II: Chord prediction accuracy

Root pitch accuracy. We investigate *Melody*'s root pitch prediction with the confusion matrix in Fig. 1, which is normalized to 100 by the row and tells which predictions are common for each target chord. The histogram on the y-axis represents chord occurrence in test set. F is often predicted instead of B, however B itself is a rare chord. Among the highest occurrence chords, no particular error that stands out. Confusion matrices for triad and added note can be found in the appendix A.

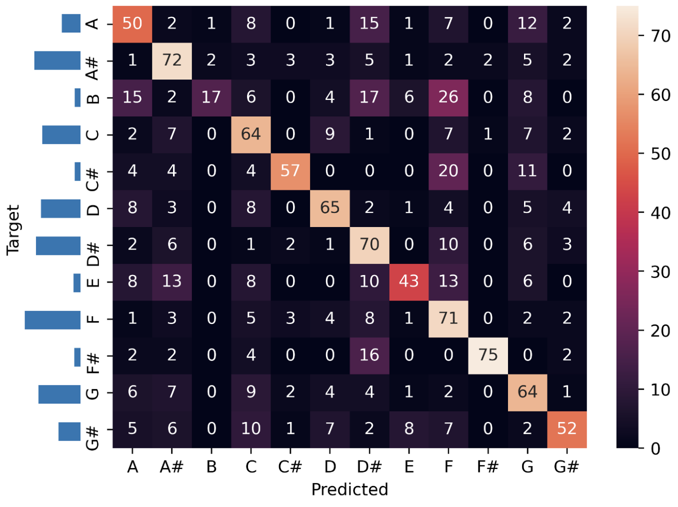

Figure 1: Root pitch confusion matrix for the Melody model, values in percent.

Previous chord prediction. We observe that our models perform better when the previous chord prediction was correct, as seen in Table III. This is aligned with the workings of an LSTM. A correct prediction indicates that the internal state of the LSTM represents accurately the song, thus the next prediction is more likely to be correct. Interestingly, *Melody* improves only in the case of previous prediction incorrect. That suggests that the melody information is helping the model to correct its internal state to a better representation of the song.

Model	prev. correct	prev. incorrect
Baseline	65.77	35.48
Melody	65.30	40.61

Table III: Accuracy depending on previous chord prediction

Additional experiments. We conduct additional experiments that include F-score depending on chord occurrence and accuracy in prediction depending on sequences of pairs of chords. However, a general insightful trend is not observed in the results, which can be found in Appendices C and D.

V. Song case studies

In a few cases, we could find a clear pattern where the notes played during the previous chord corresponds to the notes of the target chord. This helps the *Melody* model to rightly predict some chords that the baseline could not predict. The baseline could predict the root note most of the time, but not the complete chord. This improvement is best observable when looking at the accuracy for predicting the major chords.

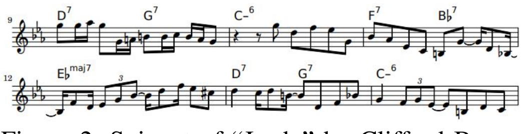

Figure 2: Snippet of "Jordu" by Clifford Brown

When looking at the played notes during the chord $F7$ that is before the chord $B\flat 7$ ($A\sharp 7$ in our mapping), we have $B\flat, A\flat, E\flat$ and C. The 4 notes of the $B\flat 7$ chord are $B\flat, D, F$ and $A\flat$. Hence, in this case the previous melody contained the root and extension pitches of the chord.

Similarly, B, G, D and $B\flat$ are played previous to $E\flat$maj7 which consists of $E\flat, G, B\flat$ and D.

Both examples show improved predictions, when comparing the *Melody* model to the *Baseline*.However, the following example shows how melody can mislead the model into making wrong predictions. Here, $F, A\flat$ and $B\flat$ were played previous to the chord change (all contained in $A\sharp 7$ chord), leading the *Melody* model to predict $A\sharp 7$ chord instead of $F7$. Without this melody information, the baseline model performed well.

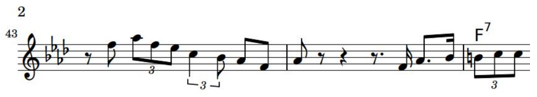

Figure 3: Snippet of "Work song" by Nat Adderley

In general, we observed that the end of the note sequences played during each chord do not tend to represent the notes played for the next Chord. This would also explain why there is no improvement between the weighted Melody models and the non-weighted. Also, the notes played during a chord tend to stronger represent the current chord than the next chord.

VI. Conclusions

In conclusion, we have trained models that are able to predict chords with relative success, achieving 53.93% of accuracy. We observed a trend that leveraging melody information can improve chord prediction, in our case by 3%. However, due to the small size of the test set, our results can be seen as preliminary, more thorough experiments would need to be conducted. Furthermore, accuracy is a very raw measure and does not capture how musically close a wrong prediction is to the target. To overcome this, we looked at confusion matrices and conducted a qualitative case study of selected songs. While we found some interesting behaviors, we did not see a clear pattern on how the melody affects the prediction. Future work could explore whether encoding melody sequentially gives more information than using a bag of notes.

VII. Acknowledgments

This study was done in accordance with the Digital and Cognitive Musicology Lab from EPFL. We want to use this opportunity to thank our supervisors Harasim Daniel and Mc Leod Andrew Philip which were always there to answer our questions and gave us valuable feedback during the entirety of this project.

References

[1] M. Pfleiderer, K. Frieler, J. Abeßer, W.-G. Zaddach, and B. Burkhart, Eds., *Inside the Jazzomat - New Perspectives for Jazz Research*. Schott Campus, 2017.

[2] S. H. Hakimi, N. Bhonker, and R. El-Yaniv, *BebopNet: Deep Neural Models for Personalized Jazz Improvisations*, 2020.

[3] D. Eck and J. Schmidhuber, *Finding temporal structure in music: blues improvisation with LSTM recurrent networks*, 2002.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

Appendix A.
Triad form and chord extension confusion matrix

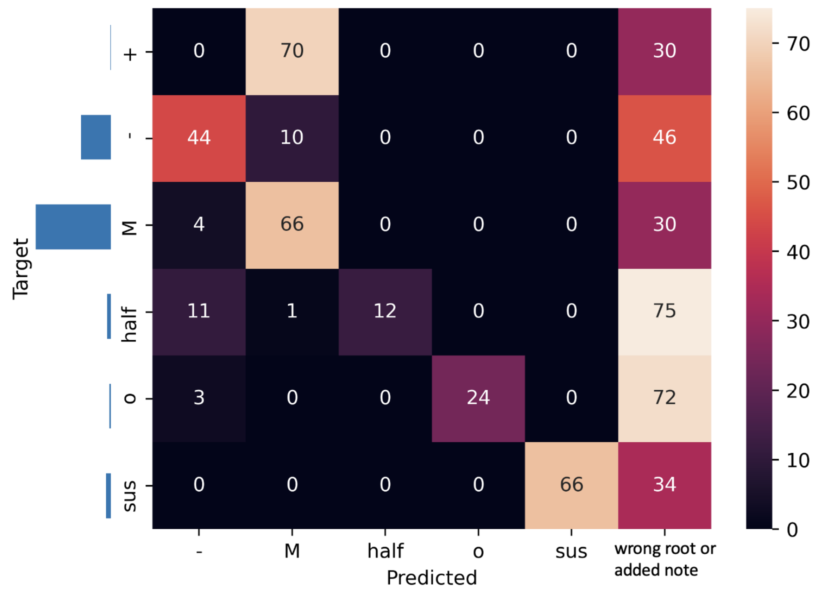

Figure 4: Triad confusion matrix for the Melody model, values in percent.

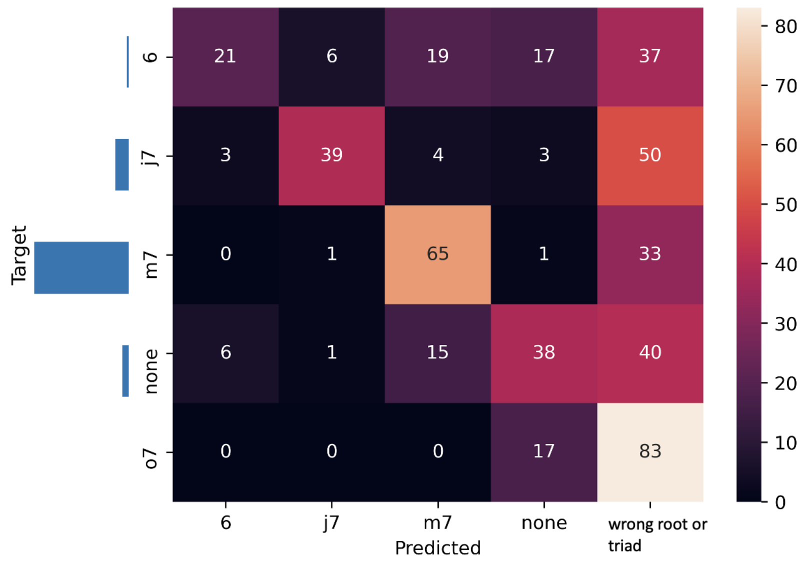

Figure 5: Added note confusion matrix for the Melody model, values in percent.

APPENDIX B.
MAPPED CHORD VOCABULARY FOR C ROOT PITCH. IDEM FOR THE REST.

Ref.	Map
C	C
C/A	C
C/A♭	C
C/B	C
C/B♭	C
C/C	C
C/D	C
C/E	C
C/E♭	C
C/F	C
C/F♯	C
C/G	C
C+	C+
C+/A♭	C+
C+/C	C+
C+7	C+7
C+79	C+7
C+79♯	C+7
C+7911♯	C+7
C+79♭	C+7
C+j7	C+j7
C+j7/C	C+j7
C-	C-
C-/A♭	C-
C-/B♭	C-
C-/C	C-
C-/C♯	C-
C-/D♭	C-
C-/E♭	C-
C-/F	C-
C-/G	C-
C-6	C-6
C-6/C	C-6
C-6/E♭	C-6
C-69	C-6
C-69/A♭	C-6
C-69/D♭	C-6
C-7	C-7
C-7/A♭	C-7
C-7/B♭	C-7
C-7/C	C-7
C-7/D	C-7
C-7/E♭	C-7
C-7/F	C-7
C-7/G♭	C-7
C-79	C-7
C-79/A♭	C-7
C-7911	C-7
C-7913	C-7
C-79♭	C-7
Cm7b5	C-7b5
C-j7	C-j7
C-j7911♯	C-j7
C-j7913	C-j7
C6	C6
C6/A	C6
C6/B♭	C6
C6/C	C6
C6/D	C6
C6/G	C6
C69	C6
C6911♯	C6
C7	C7
C7/A	C7
C7/A♭	C7
C7/B	C7
C7/C	C7
C7/D	C7
C7/D♯	C7
C7/E	C7
C7/E♭	C7
C7/F♯	C7
C7/G	C7
C79	C7
C79♯	C7
C79♯11♯	C7
C79♯13	C7
C79/A♭	C7
C79/C	C7
C7911	C7
C7911♯	C7
C7913	C7
C7913♭	C7
C79♭	C7
C79♭/B	C7
C79♭13	C7
C79♭13♭	C7
C7alt	C7
Cj7	Cj7
Cj7/A	Cj7
Cj7/A♯	Cj7
Cj7/B	Cj7
Cj7/D♯	Cj7
Cj79	Cj7
Cj79♯	Cj7
Cj79♯11♯	Cj7
Cj7911♯	Cj7
Cj7911♯/A♭	Cj7
Cj7911♯/C	Cj7
Co	Co
Co7	Co7
Csus	Csus
Csus/A	Csus
Csus7	Csus7
Csus7/A	Csus7
Csus79	Csus7
Csus7913	Csus7

Appendix C.
F-score

The best performing model, *Melody*, was further analysed by computing the F-score for each chord, and the result weighted using the chord appearance in the training dataset. The results are presented in Fig. 6, where only chords appearing more than 10 times in the test set are considered (for a more robust F-score calculation). Chords appearing more than 900 times in the train set have an F-score higher than 0.39, but they share the upper F-score range with chords showing less appearance in the training set. Hence we can't conclude that there is a linear relationship between the F-score of the chords and their sample size in the training set.

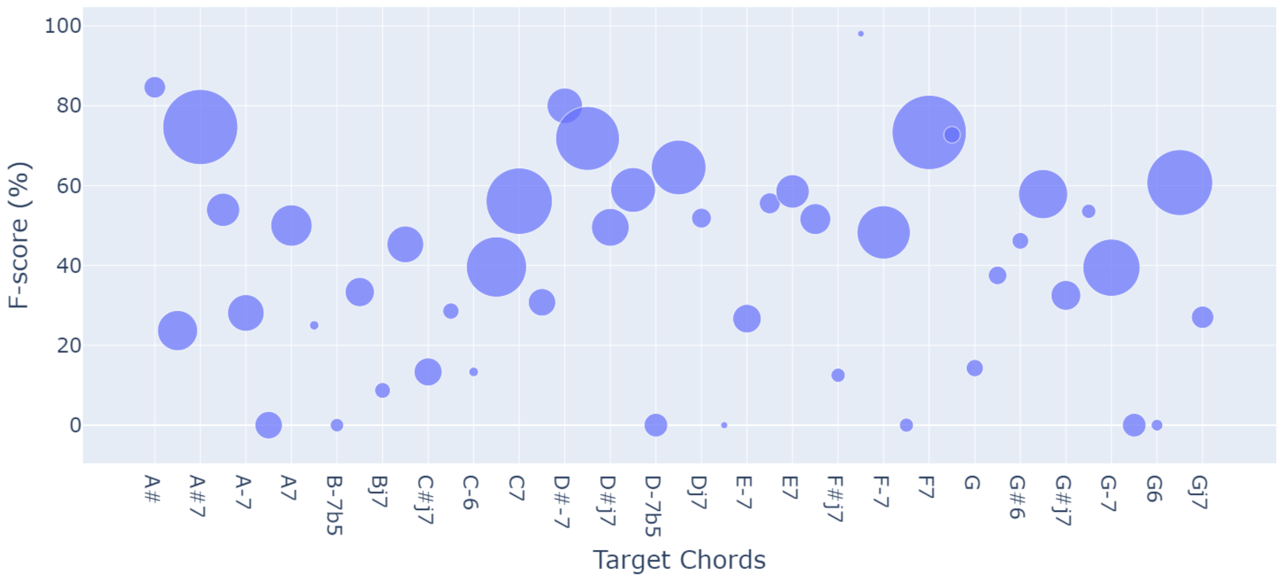

Figure 6: F-score per chord for the melody model

Appendix D. Chord sequence analysis

By analysing the chord accuracy depending on the frequency of a chord sequence in the training set (Fig. 7), defined as the previous target followed by the target to predict, we observed that in the baseline model the chord accuracy correlates with the frequency for sequences appearing between 50 and 300 times in training, then stabilises. However, in the melody model this correlation is not so clear when looking at low frequency sequences, meaning the melody may have been helpful in those cases.

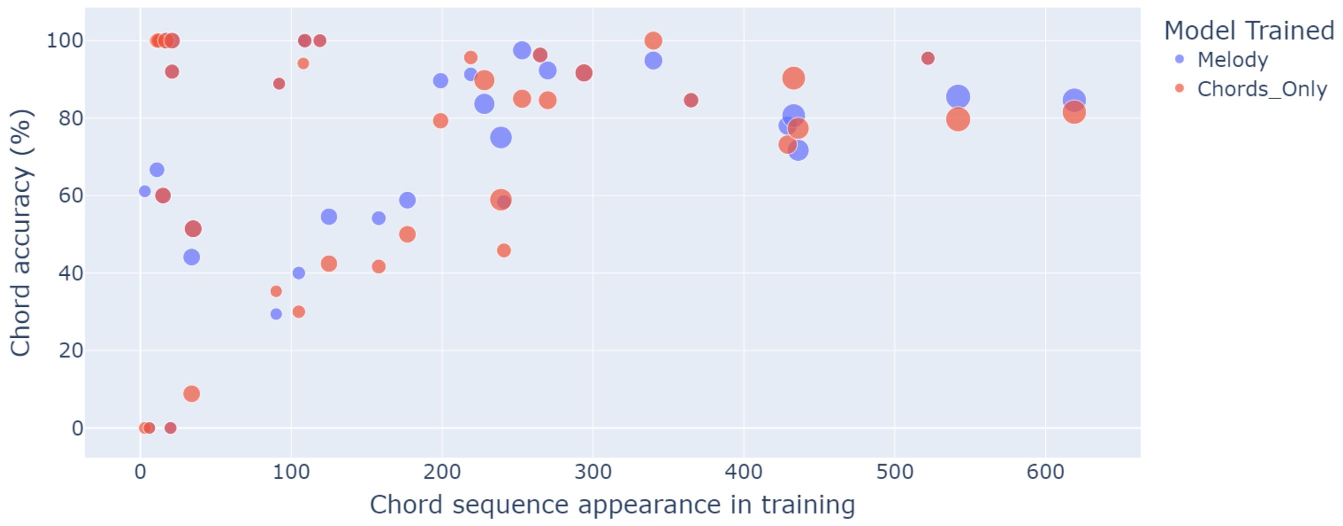

Figure 7: Chord accuracy vs chord sequence frequency in the training set