Modeling Piano Fingering Decisions with Conditional Random Fields

David A. Randolph
Department of Computer Science
University of Illinois Chicago
drando2@uic.edu

Barbara Di Eugenio
Department of Computer Science
University of Illinois Chicago
bdieugen@uic.edu

Justin Badgerow
Department of Music
Elizabethtown College
badgerowj@etown.edu

ABSTRACT

Deciding what fingerings to use is a core skill for accomplished pianists. We model piano fingering decisions with conditional random fields, demonstrating the power and flexibility of this approach to produce results to compete with the state of the art. We present new corpora of fingering data, compiled from professional pianists and editorial scores. We analyze recently suggested metrics for evaluating fingering systems and discuss drawbacks in their application to models that do not assume segregation of hand assignments.

1. INTRODUCTION

Deciding what fingerings to use is a core skill for accomplished pianists. However, most scores contain at best only sparse fingering annotation, and many scores provide no fingering suggestions at all. With the explosion of freely available sources of machine-readable music (as on the popular MuseScore [1] site, which as of this writing boasts over one million such digital scores for piano), automated systems to generate fingering advice promise to help pianists better prepare pieces for performance. An overview of the problem is provided in Figure 1.

The use of hidden Markov models (HMMs) as a method for modeling piano fingering decisions has been well-studied. In 2007, Yonebayashi and colleagues [4] suggest HMMs for melodic passages. Five years later, Nakamura et al. [5] expand this approach, merging HMM output for polyphonic music. This work has culminated recently with the application of higher-order HMMs to lay explicit claim to the state of the art in the domain [6], while at the same time suggesting upper limits for the HMM approach.

As HMMs are generative models, their formalism includes emission probabilities of the underlying sequence of notes. HMMs try to model the entire process of generating music, which likely overestimates the role a series of fingering decisions might have in giving rise to a series of notes. Since we are fundamentally interested only in inferring the most likely fingering sequence, much of the machinery of the HMM seems to be non-productive. The sequence of notes is a given. This suggests that a more direct (discriminative) approach may produce improved results.

[1] https://musescore.com

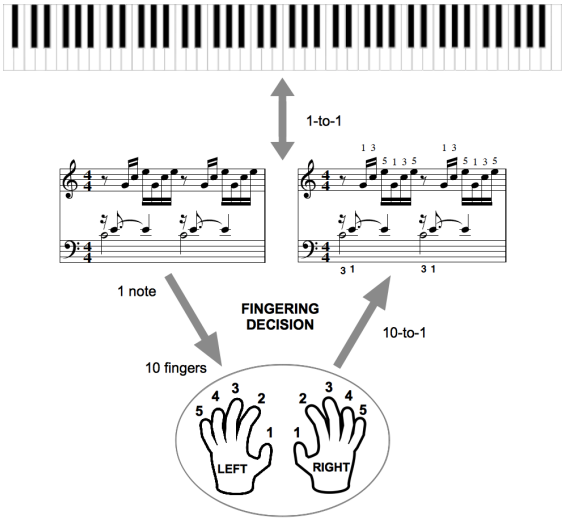

Figure 1. Overview of the piano fingering problem. *Credits: Inspired by [1], with keyboard by [2] and hand by [3].*

The most popular discriminative models are conditional random fields (CRFs). CRFs are undirected graphical models, which means reasoning about any point in a sequence of events (notes) is not constrained to a fixed number of preceding events, as is a limitation with HMMs. Indeed, CRFs implicitly support unlimited long-distance dependencies between the observed (note) states: Feature functions for any note may include features from literally any other note in a sequence. CRFs offer access similar to that provided by HMMs to hidden (fingering) states. Higher order CRFs, at additional computational cost, also allow functions to consider speculative fingerings of remote notes. This flexibility holds great promise for a rich variety of CRF models.

We here describe the first application of a CRF model to the piano fingering problem and compare its performance to the latest HMM model described in the literature [6]. We also highlight subtle limitations for evaluating system performance using match rates [6].

2. APPROACH

CRFs are often described as logistic regression for sequential data. In its most general form, the CRF looks to maximize $P(\mathbf{y}|\mathbf{X})$, the conditional probability of a label (fin-

gering) sequence $\mathbf{y}$, given an observed sequence of events (notes) $\mathbf{X}$, like so:

$$P(\mathbf{y}|\mathbf{X}) = \frac{1}{z} \exp\left(\sum_{i=1}^{N} F(\mathbf{X}, \mathbf{y}, i)\right), \quad (1)$$

where N is the length of the sequence, z is a normalizing constant, and

$$F(\mathbf{X}, \mathbf{y}, i) = \sum_{j=1}^{J} w_j f_j(\mathbf{X}, \mathbf{y}_{i-k}^{i}, i), \quad (2)$$

where J is the number feature functions f.

More specifically and pragmatically, a k^{th}-order linear chain CRF looks to maximize $P(\mathbf{y}|\mathbf{X})$ by reasoning about all observed events, but it considers labels from only the prior k events encountered. Formally,

$$P(\mathbf{y}|\mathbf{X}) = \frac{1}{z} \exp\left(\sum_{i=1}^{N} F(\mathbf{X}, \mathbf{y}_{i-k}^{i}, i)\right), \quad (3)$$

where N is the length of the sequence, z is a normalizing constant,

$$F(\mathbf{X}, \mathbf{y}_{i-k}^{i}, i) = \sum_{j} w_j f_j(\mathbf{X}, \mathbf{y}_{i-k}^{i}, i) \quad (4)$$

sums the weighted feature functions f defined for the model, j ranges over the number of feature functions, and k is a constant defining the order of the model.

In the most common "linear chain" CRF, $k = 1$, and feature functions are restricted to examining only the current and previous labels in the training example. Higher order models allow the consideration of additional elements in $\mathbf{y}$, at the cost of more computing time. The machine learning task is to infer the optimal set of weights w, applying some type of gradient descent. In this study, we consider first-order linear chain CRFs, defined formally as maximizing

$$P(\mathbf{y}|\mathbf{X}) = \frac{1}{z} \exp\left(\sum_{i=1}^{N} F(\mathbf{X}, y_i, y_{i-1}, i)\right), \quad (5)$$

where N is the length of the sequence, z is a normalizing constant, and

$$F(\mathbf{X}, y_i, y_{i-1}, i) = \sum_{j} w_j f_j(\mathbf{X}, y_i, y_{i-1}, i). \quad (6)$$

3. FEATURES

All features are defined exclusively in terms of the observed note sequence attributes $\mathbf{X}$. We implement features related to a variety of performance and musical attributes, with some (highlighted in boldface type) making their first appearance in the literature:

- physical distances between keyboard keys
- concurrent notes
- segment boundaries

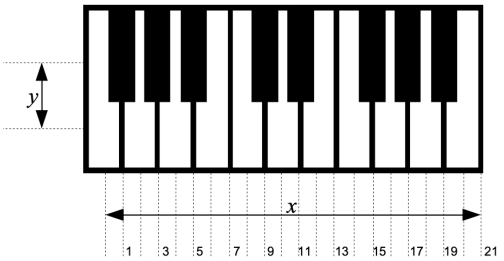

Figure 2. Visualization of lattice distance measure recommended by Nakamura and associates [6].

- black key usage
- **dynamics**
- **articulation**
- **temporal constraints**
- note repetition

3.1 Distance Features

We adopt the "lattice distance" measure between notes recommended by Nakamura et al. [6], which includes both a horizontal and vertical component. (See Figure 2.) Moving horizontally from the center of any key, one unit of distance is added when reaching the center of a white key or a gap between two white keys. Vertically, one unit is recorded if the transition is to a differently colored key. Distances are positive moving right or up and negative moving left or down. We maintain the measures separately. (We do not attempt to combine the x and y dimensions, though this might be done to achieve a single estimate of physical distance.) We also institute their recommended cap on a maximum absolute horizontal "leap" distance between notes. At some point, the hand must be radically re-positioned, and the specific magnitude of this move is irrelevant; tracking the exact magnitude of the move becomes noise to the model. We therefore institute a ceiling on the horizontal distance of 15 units, slightly more than an octave.

For a given note in a sequence, we capture both horizontal and vertical distances between it and each of the three notes preceding and following it, for a total of 12 numerical features per note. We then concatenate the horizontal distances within three-, five-, and seven-note envelopes centered on the current note to define three categorical "n-gram" features, which we speculate will capture more explicitly the contour of notes in the neighborhood.

We also capture whether a leap, as defined above, has been detected between the current and previous note.

In passing, we observe that using pitches and pitch n-grams directly, as is often seen done with words in superficially similar natural language processing tasks, appears less effective here. The n-grams might be at a disadvantage compared to using a distance measure, as they provide fewer opportunities to learn from isomorphic situations (or "transposition symmetries," as Nakamura et al. might call them).

3.2 Concurrency Features

We implement code to detect that one or more note onsets are coincident with the current note—that is, that a chord has been struck. Specifically, we add one function to count the number of coincident note attacks with pitches lower than the pitch at index i and one to count notes with higher pitches. Concurrency is detected for each function when the onset times are within 30 ms of the indexed note, a quantization value established empirically [12] and applied previously by Nakamura and associates [6]. These higher and lower note counts are concatenated along with the staff name into a categorical feature.

As we impose a total order for sequences that sorts first by temporal order and then by ascending pitch, notes involved in a chord require fingers that follow the order of notes (ascending for the right hand and descending for the left).

We also institute a feature to detect a chord border. If either of the two upper or lower coincident attack count are zero and the other is not, we flag a chord border. We perform similar actions to detect overlapping notes based on note release times, contemplating a more general "vertical cost" suggested by Al Kasimi et al. in 2007 [13]. We assemble counts of notes higher and lower than the current note that overlap its attack by more than 30 ms. Assuming no damper pedal is in use, this should convey constraints on fingering choice for moving notes.

3.3 Segment Boundary Features

These are simple features to indicate the beginning and end of a note sequence.

3.4 Black Key Features

We include three boolean unigram features to indicate if the current, preceding, and following notes are on black keys and a trigram categorical feature that concatenates the contour of black keys within this envelope.

3.5 Dynamic Features

How hard a key needs to be struck can affect fingering choices. We therefore include the MIDI velocity number of the current note as a feature. Including n-gram velocity features are likely also viable, but meaningful data with high variance is lacking for dynamics in the datasets currently available. To our knowledge, this is the first application of such a feature in the literature.

3.6 Articulation Features

Many expert systems have constrained their models to assume all notes are played *finger legato*, in a smooth connected way. Such playing emphasizes fast in-position finger transitions and pivots crossing fingers over and under the thumb. As we aim for more general predictive power, we attempt to define and to quantify *legato* and *staccato* articulation in our features.

From a segregated fingering standpoint at least, staccato is a purely melodic phenomenon. We are looking to capture when finger order does not align with note order (where pivoting occurs without the thumb). When chords are sounding, the hand is anchored and is not free to perform such feats. Therefore, we look for a melodic window around the note and calculate measures of separation between notes within the window. The three articulation features captured at every sequence position are defined as follows:

- Staccato count: the number of notes in the 9-note window surrounding the current note that are followed by silence at least half as long as the note itself
- Normalized silence: the sum of inter-note silence surrounding the current note, divided by the total duration of the 9-note window, measured from the leftmost note onset to the rightmost onset
- Separated note count: the number of notes surrounding the current note that are separated by at least 60 milliseconds

This is the first explicit use of this type of performance data among published models.

3.7 Temporal Features

In another first, we capture three continuous features to estimate the rate of passing notes at a given point in the sequence. This is not an inference of an overall tempo marking in a score like *Allegro* or *Andante* or a metronome marking, though we do consider metronome changes detected in the *music21* representation, as these changes affect attack times and note duration. We simply calculate the rate of note attacks per second for the five notes prior to and including the current note. Then we repeat this for future notes, and finally throughout the entire 9-note window. (When playing a 64th-note run, a pianist is perhaps less convinced of the slowness of a *Larghissimo* piece. The marked tempo of a piece is an indication of pulse and not necessarily the rate of sounding notes.)

3.8 Repetition Features

Finally, we record how many times the same note has been struck immediately before and after the current note in the sequence.

4. CORPUS DEVELOPMENT

In this section, we introduce two new corpora created for the piano fingering task and a previously existing corpus we have transformed for our use here.

All pieces included in the new corpora are segmented into phrases and are made freely available for use by other researchers at `https://github.com/dvdrndlph/didactyl`. Note that phrase segmentation information is not leveraged by the CRF models that are the focus of this paper.

4.1 The Layer One Corpus

We present a new corpus based on six popular intermediate sonatinas written by Muzio Clementi (Opus 36) circa

1797, noted for their progressively difficulty. The availability of MusicXML transcriptions of these works at MuseScore [14] and the *xml2abc* utility [15] simplified the task of creating high-quality scores in the abc format preferred by the *abcDE* editor [16], which we elected to use for annotation. For convenience, we divided each sonatina manually into smaller pieces: each section demarcated by double bar-lines was assigned to a separate file. This left us with 27 distinct files in the corpus.

The five editorial scores with fingerings [17–21] are found in the public domain and are readily available on IMSLP [22], along with the fingerings proposed by the composer [23], provided easy access to (albeit incomplete) expert fingering data. Using *abcDE*, we captured these data in a set of abcD files. Through this activity, several obvious errors were corrected in the note transcriptions.

4.1.1 Annotating Phrases

In their work on stringed instrument fingering, Radicioni et al. [24] argue convincingly that segmented (localized) fingering solutions lead to improved global solutions. Citing [25–27], they stress that "expressive aspects of performance descend from the performer's analysis of musical [structure]," especially phrases, and that fingering is one such aspect. Since many piano fingering models focus exclusively on melodic passages, and automatic phrase segmentation remains an open problem, we elected to include explicit phrase separation in the Layer One corpus. This is the same approach employed by Radicioni and associates [24], allowing phrase segmentation to be provided as an additional input to fingering models.

Our two pianist collaborators agree that this "divide-and-conquer" technique is familiar and natural: pieces are segmented into phrases, and the phrases are fingered individually, more or less in isolation. Admittedly, how they internalize the definition of *phrase* is fraught, but musicians do use this term as if it were broadly understood. Conductors certainly use it, and rehearsals rarely devolve into semantic arguments.

Clearly, there may be considerable variability in where different pianists, accustomed to a high degree of autonomy in such matters, draw the lines between segments, likely reflecting tension between conflicting definitions (relating to prosody, affect, or basic sectioning), among other factors. All of which provides further motivation to include phrasing information as part of the corpora. For now, a phrase, as we discuss in Section 4.1.2 below, is whatever our collaborating pianists agree it is.

4.1.2 Phrase Agreement Experiment

We asked two advanced pianists (both of whom have advanced degrees in piano performance and over 20 years of professional experience teaching and playing piano) to segment five sections from Clementi's *Six Sonatinas* (Op. 36) into phrases. Our pianists agreed before doing the segmentation work that these pieces should have "obvious phrase boundaries." Specifically, we used sections ending at the first double-bar line in Sonatinas 2–6. The pianists were provided very simple annotation guidelines:

> Mark the notes that end complete musical thoughts, typically supported by the presence of a cadence. Each voice in a piano score will have its own independent phrasing.

Prior to annotating the five selections, the pianists actually annotated two sections from Sonatina 1. Disagreements were discussed, and perfect consensus was achieved. Over the five test selections, however, we have mixed results. The task amounts to binary labeling of each note as marking the end of a phrase or not. For the upper staff notes, we see excellent agreement: on 1255 of 1262 notes, our pianists apply the same label (Cohen's $\kappa \approx 0.879$). For the lower staff, however, the agreement is less substantial: 914 of 945 notes ($\kappa \approx 0.410$). After analyzing the discrepancies, we concluded that voice independence was interpreted differently by the annotators. To encourage better agreement, we modified our guidelines to be more nuanced:

> The primary task is to demarcate phrases in the score. Mark the notes that end complete musical thoughts, typically supported by the presence of a cadence.
>
> Each voice in a piano score *may* have its own independent phrasing. However, when a lower voice is accompanying an upper voice, the lower will typically end a phrase around the same time as the upper, coordinating to create the sense of cadence. Pay special attention when deviating from this general rule. . . .

But this did not help agreement in the lower staff over our test set. Since Radicioni and associates [24] present their ideas in the context of unaccompanied melodic lines and do not discuss phrasing in a homophonic context, it may be that such accompanying voices that frequent the lower staff of piano scores are less amenable to phrase segmentation.

4.1.3 Layer One Corpus Summary

In sum, we have partial fingering data for the first sections of all six sonatinas (the "Layer One" corpus) from six different editorial sources and complete fingering from six professional pianists for these same selections. All sections include phrase segmentation, which may be used as features or even alternative experimental units.

Only the completely annotated examples, elements in the "Layer One Full" ($\mathrm{L1_{full}}$) subset, are usable in the experiments described below.

4.2 The Beringer Corpus

After transcribing numerous exercises from Beringer and Dunhill's *Manual of Scales, Arpeggios, and Broken Chords for Pianoforte* [28] to abc format, we used the abcDE annotation editor [16] to record all of Beringer's suggested fingerings, including alternates, in abcDF format. Since such technical exercises form a large part of the daily practice regimen of many pianists, it seems logical to have such standard fingerings on hand to train various models. A

good model should presumably practice the way humans do.

All exercises, except chromatic scales, are included for both hands and in all 12 keys. Three major scale exercises (2088 notes total), three harmonic minor scales (2088), six melodic minor scales (4176), one major arpeggio exercise (936), one minor arpeggio (936), one major broken chord exercise (768), and one minor chord (768) are present, along with two chromatic scales (100). This totals 11,860 notes over 194 exercises. All exercises are segmented into phrases, segments clearly signaled in the score with double bar lines.

We observe that not all notes are explicitly annotated by Beringer. However, the assumed editorial and *pedagogical* intent is to specify the fingerings of each exercise completely and unambiguously. To confirm that annotations are only removed to reduce clutter and introduce no ambiguity, we asked two experienced pianists to provide the missing fingerings, which they did with perfect agreement.

4.3 The PIG Dataset

Nakamura et al. [6] have released a large corpus known as the PIG Dataset.[2] Composed of opening fragments from 150 intermediate and advanced works from the standard piano repertoire. It defines a special test set of 10 selections each by Bach, Mozart, and Chopin, representing the Baroque, Classical, and Romantic periods of Western music. Each selection in the test set is associated with fingering sequences from four (Bach), five (Chopin), or six (Mozart) different pianists. The remaining selections, comprising the putative training set, come from a variety of composers and are annotated by one or two different pianists.

They leverage this corpus to stake a claim for their third-order HMM as the state of the art. This is a significant milestone, as we finally have sufficiently large corpora to make such claims reasonable and defensible. It seems likely to become standard practice for new models to describe how they perform with respect to this corpus and in terms of the evaluation metrics Nakamura and associates have put forward. We have already seen this for models from Guan and associates [29, 30].

5. EXPERIMENTS

5.1 Implementation

We implement our first-order linear chain CRF model leveraging the *sklearn-crfsuite* Python module [7], training via its *lbfgs* limited-memory approximation of the Broyden-Fletcher-Goldfarb-Shanno algorithm [8–11]. For the initial experiments described here, we accept all default hyperparameter settings for *lbfgs* ($c_1 = 0$, $c_2 = 1$, $\varepsilon = 10^{-5}$, $\delta = 10^{-5}$, and $period = 10$). We perform no hyperparameter optimization or feature selection preprocessing. All specific features described in 3 below are included in the model and are weighted exclusively by the *lbfgs* training algorithm. By skipping these steps, we establish a lower bound for the performance of the CRF approach.

To optimize performance, *sklearn-crfsuite* does not support true feature functions of both $\mathbf{X}$ and $\mathbf{y}$ values. Instead, it combines each feature with all possible tag (fingering) transitions to form its fully constituted feature functions. That is, $f_j(\mathbf{X}, y_i, y_{i-1}, i)$ are rendered internally as conjunctions of independent functions of $\mathbf{X}$ and $\mathbf{y}_{i-1}^{i}$:

$$F(\mathbf{X}, \mathbf{y}_{i-1}^{i}, i) \sim \sum_j \left[w_j f_j(\mathbf{X}, i) \prod_t \prod_u g_{t,u}(\mathbf{y}_{i-1}^{i}, i) \right], \tag{7}$$

where t and u range over the set of possible tags and $g_{t,u}$ is a binary indicator function for when both tags t and u appear sequentially (when $y_{i-1} = t$ and $y_i = u$).

5.2 Methodology

Except for evaluations performed with PIG test data, we perform five-fold cross-validation of our CRF model using various subsets of the corpora described above. Specifically, each subset is partitioned into five random folds of data, grouped to ensure that no data from the same musical piece is included in more than one fold. This is done to avoid over-fitting, as even having fragments from one piece in both training and test sets can artificially inflate estimates of model performance, given the repetitive nature of music. Since this procedure almost certainly produces unbalanced folds in terms of note counts, we calculate weighted averages of the various metrics used. (This grouping naturally occurs if and only if exactly one ground truth is provided, and the unit to be evaluated is an entire piece. However, we adopt this as a standard practice, generally applicable to both complete and more granularly segmented data, with any number of annotators.) Cross-validation is used to reduce variance in model evaluation.

We use *scikit-learn*'s `KGroupFold` method to split the data into five groups. Because this method does not support random seeding, we randomly order each subset before splitting to minimize any dependencies between examples. In turn, each group is held out, the model is trained with the remaining data, and it is evaluated over the held-out data. We calculate total accuracy and F_1 score (at the note level) and report the weighted average of the results for each fold.

We also apply three of the match rates suggested by Nakamura et al. [6] in two different ways. They define a "match rate" for a score (or "ground truth") as simply "the fraction of notes for which the estimated fingerings are correct." They say the "general match rate" (M_{gen}) represents "how closely an estimate agrees with all the ground truths," which must imply an average match rate across all ground truths. We further infer that M_{gen} for an entire set of scores, must be an average of these averages. Weighting this average by the number of notes in each score would allso seem to be appropriate, but it is unclear if this is done in the results reported in the original paper. Both weighted and unweighted M scores are reported here.

Nakamura et al. [6] also suggest a "highest match rate" (M_{high}) measure, which accepts the maximum match rate

[2] Available at `https://beam.kisarazu.ac.jp/~saito/research/PianoFingeringDataset/`.

seen for any ground truth, and a "soft-match rate" (M_{soft}) measure, which gives credit for a match if a predicted finger matches any ground truth. Neither of these measures are averages at the score level, but they too are potentially subject to weighting when they are averaged to generate a score for an entire data set.

The results of these experiments are summarized in Table 1, which also provides size estimates for the data subsets. **Segments** actually reflect the channel or staff separation, which is available in both symbolic and most MIDI representations of piano music. For our purposes here, there are two segments in each score. Each **Tag** represents an annotated note. (If there are 10 notes in each segment and three different pianists providing annotations, this one score would contribute 60 tags to the dataset.)

5.3 New Corpus Experiments

Our initial experiments involve working with our own corpora. Notable is the model's good performance with the "rudimentary" subsets of the Beringer corpus. It is gratifying to see the model is able to predict long-accepted standard fingerings reasonably well. Its struggles with the L1_{full} subset are disappointing, but perhaps understandable considering the diversity of fingering advice across the corpus. The model receives as many mixed messages in training as it does in testing. We also note an unexpectedly large number of non-default fingerings in the corpus.

The most noteworthy result here is how combining the rudiments from Beringer corpora with L1_{full} data produced results very close to the higher performing rudimentary corpus. Unweighted M_{gen} for L1_{full} data and rudiments together is over 29 points higher than that for L1_{full} alone. Some of this gain is attributable to the higher number, and shorter length, of rudimentary segments in the data set, but when segment length is controlled for in the weighted M_{gen} score, the increase is still 24 points. This seems to indicate that the CRF model is capable of generalizing what it learns from the rudiments to the more musical examples in the combined data set. This should be gratifying to piano teachers who have long preached the value practicing scales and arpeggios. If replicable, this result could also have profound effects on how future models should be trained.

Encouraged by these results, we proceed to test our model's performance over larger and more diverse datasets.

5.4 PIG Dataset Experiments

5.4.1 PIG Data Transformation

Our CRF model is built on top of *Pydactyl* [32], which makes heavy use of the *music21* Python module [33] and leverages its highly refined *score* representation of music.

Each PIG dataset file, on the other hand, is a text-based "piano roll representation," with highly granular note-on and note-off timestamps, measured in seconds from the beginning of the piece. While it seems none of the event data is from an actual human performance (given the identical note onsets for notes in chords), it is still impossible for *music21* to interpret such a file without loss of fidelity. The shortest note in *music21* is a 1/2048th note, but this has only a relative relationship to actual duration in seconds.

Another issue is the ordering of notes in PIG files. The events are sorted by note onset time, but there is no apparent secondary sorting elements. The order of notes from left-to-right through time and then from low-to-high by pitch is essential to how abcD keeps fingerings aligned with notes. Therefore, it is necessary to impose this total order requirement and accept some deviations in the exact timing of notes when transforming PIG files to abcD.

We have developed reliable code to perform this transform with help from the Mido Python package [34], and an implementation for the opposite transformation, from abcD to PIG format, is forthcoming, as we hope to maximize interoperability.

5.4.2 PIG Training Set Experiment

Our first PIG experiment is simply to run five-fold cross-validation across the training subset of the PIG Dataset. This demonstrates a weighted mean fold accuracy of 66.88%, a mean fold F_1 score of 66.18%, and a weighted and unweighted M_{gen} scores of 67.07% and 66.40%, respectively. This is encouraging, as the published M_{gen} result from Nakamura et al. [6] is 64.5% for their third-order HMM. Complete results are in Table 1.

5.4.3 PIG Dataset Experiments

We train our model with the PIG training data and evaluate it with the PIG Test data, as is done by Nakamura and associates [6]. This is to allow a head-to-head comparison of the performance of the two models. The results, summarized in Table 2, are mixed. While the CRF holds a slim advantage per the weighted and unweighted M_{gen} measures, the HMM holds advantages between 0.004 and 0.025 according to the other, more relaxed, metrics.

We also performed five-fold cross-validation following the over the entire PIG Dataset and finally over all available data. The additional data produces markedly better results, with advantages between 3.2 and 9.5 percentage points for every metric, as seen at the bottom of Table 1. This suggests additional training remains beneficial. However, a final model trained with all data except the PIG test data and evaluated with same produced slightly degraded performance (of less than one point for each metric reported in Figure 1.)

6. DISCUSSION

There seems to be a significant flaw in the application of match rates to the PIG Dataset. The piano fingering problem is typically framed as the assignment of fingerings to notes in a printed score, the symbolic representation of the notes that are to be performed. Granted, this is not the only way to define the problem, but it is clearly the one that is applicable to the repertoire represented in the PIG Dataset. The problem relates to channel separation—how a note assigned to a staff. The convention within PIG is always to assign a channel/staff that is consistent with the finger used to play the note. This may seem reasonable, since they are not processing (machine-readable) symbolic

Data Set	Segments	Tags	Weighted Fold Means							
			Accuracy	$\mathbf{F_1}$	*Unweighted*			*Weighted*		
					$\mathbf{M_{gen}}$	$\mathbf{M_{high}}$	$\mathbf{M_{soft}}$	$\mathbf{M_{gen}}$	$\mathbf{M_{high}}$	$\mathbf{M_{soft}}$
Scale	164	18870	0.6916	0.6913	0.7074	0.7346	0.7518	0.7014	0.7229	0.7390
Arpeggio	124	6344	0.8584	0.8579	0.8767	0.8798	0.8833	0.8818	0.8843	0.8879
Broken Chord	76	2432	0.8549	0.8547	0.8741	0.9285	0.9668	0.8741	0.9285	0.9668
Beringer	364	27646	0.7412	0.7402	0.8073	0.8258	0.8451	0.7697	0.7865	0.8017
$\mathrm{L1_{full}}$	96	18736	0.5089	0.4968	0.5065	0.5497	0.6399	0.5089	0.5517	0.6430
Beringer + $\mathrm{L1_{full}}$	460	46382	0.6775	0.6763	0.7979	0.8147	0.8411	0.7434	0.7600	0.7907
PIG Training	318	50026	0.6688	0.6618	0.6640	0.6707	0.7011	0.6707	0.6770	0.7035
PIG	618	100040	0.6683	0.6641	0.6691	0.6841	0.7407	0.6750	0.6893	0.7436
All	1078	146422	0.7185	0.7162	0.7644	0.7755	0.7963	0.7350	0.7466	0.7752

Table 1. Five-fold cross-validation of CRF model using various evaluation methods and subsets of available corpora.

Weight	Measure	CRF	HMM	Diff.
	Accuracy	0.6540	N/A	N/A
	F_1	0.6507	N/A	N/A
Weighted	M_{gen}	0.6501	0.645	0.005
	M_{high}	0.6869	0.710	-0.023
	M_{soft}	0.8495	0.855	-0.006
Unweighted	M_{gen}	0.6455	0.645	0.001
	M_{high}	0.6849	0.710	-0.025
	M_{soft}	0.8515	0.855	-0.004

Table 2. Comparison of our CRF model to the HMM from Nakamura et al. [6], using PIG training and test datasets.

representations. Their model operates on data very similar to a MIDI event stream. However, applying match rates as they do presuppose a single piece with multiple associated fingerings, and in some cases in the PIG Dataset, there is no single representation of the score. There are multiple. But only one of these representations, chosen arbitrarily, is presented to the model for prediction. This is problematic.

This highlights a significant difference between the Layer One corpus and the PIG Dataset: in Layer One, the channel assignment is implicit in the underlying score. It also reflects a significant difference in how the two models natively represent the problem space. Notably, both models expect some sort of channel information *as inputs*. For our CRF, this is a staff assignment, clearly an implicit attribute of conventional musical notation. But for the HMM, it is an explicit hand assignment.

Applying these match-rate metrics seems completely valid only in cases where hand assignments are completely non-controversial, where right-hand fingerings are segregated to the upper staff and left-hand fingerings to the lower.

7. CONCLUSIONS AND FUTURE WORK

We have described a straightforward first-order CRF model for predicting piano fingering decisions. It has demonstrated performance comparable to that of a competing third-order HMM system, and we contend that the flexibility and discriminative nature of CRFs make them a better choice for this domain, over more rigid generative models. We have also described new corpora which may now be leveraged in the domain.

Simply augmenting the training data further could improve system performance. But more valuably, the various features of the model may be augmented and/or enhanced. Placing limits, as is done with the "leap" ceiling for distance features, should be considered on all continuous features. All features currently operate implicitly on a single staff (and with one default hand in mind). Features that consider complete vertical slices of notes are needed. We also envision experiments with higher-level musical concepts, such as the perceived chords or keys operating among the notes surrounding each fingering decision. The *music21* library contains a wealth of possible features.

Automated feature selection and hyperparameter tuning of the CRF are clearly in order, and we have already performed promising preliminary experiments with higher-order CRFs, using the *PySeqLab* package [35]. Finally, more experiments are needed to confirm the strikingly positive impact of rudimentary training data on model performance observed in this study.

8. REFERENCES

[1] Y. Yonebayashi, H. Kameoka, and S. Sagayama, "Overview of our IJCAI-07 presentation on "Automatic Decision of Piano Fingering Based on Hidden Markov Models"," 2007. [Online]. Available: http://hil.t.u-tokyo.ac.jp/research/introduction/PianoFingering/Yonebayashi2007IJCAI-article/

[2] Clker.com, "Clker.com: Free Clipart," http://www.clker.com, Accessed: 2023-03-18.

[3] R. Dell, "Hand," https://thenounproject.com/term/hand/38129, Copyright information: CC-BY 3.0 license (https://creativecommons.org/licenses/by/3.0). Accessed: 2017-05-07.

[4] Y. Yonebayashi, H. Kameoka, and S. Sagayama, "Automatic decision of piano fingering based on a hidden Markov models," in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007, pp. 2915–2921.

[5] E. Nakamura, N. Ono, and S. Sagayama, "Merged-output HMM for piano fingering of both hands," in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2014, pp. 531–536.

[6] E. Nakamura, Y. Saito, and K. Yoshii, "Statistical learning and estimation of piano fingering," *Information Sciences*, vol. 517, pp. 68–85, 2020.

[7] M. Korobov, "sklearn-crfsuite," 2017. [Online]. Available: https://sklearn-crfsuite.readthedocs.io

[8] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms 1. general considerations," *IMA Journal of Applied Mathematics*, vol. 6, pp. 76–90, 3 1970.

[9] R. Fletcher, "A new approach to variable metric algorithms," *The Computer Journal*, vol. 13, pp. 317–322, 1970.

[10] D. Goldfarb, "A family of variable-metric methods derived by variational means," *Mathematics of Computation*, vol. 24, pp. 23–26, 1970.

[11] D. F. Shanno, "Conditioning of quasi-newton methods for function minimization," *Mathematics of Computation*, vol. 24, pp. 647–656, 1970.

[12] E. Nakamura, T. Nakamura, Y. Saito, N. Ono, and S. Sagayama, "Outer-product hidden Markov model and polyphonic MIDI score following," *Journal of New Music Research*, vol. 43, no. 2, pp. 183–201, 2014.

[13] A. Al Kasimi, E. Nichols, and C. Raphael, "A simple algorithm for automatic generation of polyphonic piano fingerings," S. Dixon, D. Bainbridge, and R. Typke, Eds., 2007, pp. 355–356.

[14] M. Clementi, "Clementi: Six Sonatinas for Piano, Op. 36," https://musescore.com/user/9292486/scores/4281241, 2017, Edited by PianoXML. [Online]. Available: https://musescore.com/user/9292486/scores/4281241

[15] W. Vree, "xml2abc," http://wim.vree.org/svgParse/xml2abc.html, accessed: 2019-02-09.

[16] D. A. Randolph and B. Di Eugenio, "Easy as abcDE: Piano fingering transcription online," in *Extended Abstracts for the Late-Breaking Demo Session of the 17th International Society for Music Information Retrieval Conference*, New York, 2016.

[17] M. Clementi, *Sonatines pour Piano à 2 mains [Sonatinas for Piano with Two Hands]*. Paris: Durand & Cie, 1890.

[18] ——, *Sei Sonatine [Six Sonatinas]*. Gio. Ricordi, n.d.

[19] ——, *Six Sonatinas for the Piano*, L. Köehler, Ed. New York: G. Schirmer, 1904, Edited by Louis Köehler.

[20] ——, *12 Sonatine [12 Sonatinas]*, B. Mugellini, Ed. Ricordi, 1904, Edited by Bruno Mugellini.

[21] ——, *Sechs Sonatinen [Six Sonatinas]*, J. J. Riefenstahl, Ed., Berlin, 1968, Edited by J. J. Riefenstahl.

[22] Project Petrucci LLC, "International Music Score Library Project," http://imslp.org.

[23] M. Clementi, *Six Sonatinas, Opus 36*, W. A. Palmer, Ed. Alfred Music, 1968.

[24] D. Radicioni, L. Anselma, and V. Lombardo, "A segmentation-based prototype to compute string instruments fingering," in *Proc. of 1st Conference on Interdisciplinary Musicology*, Graz, Austria, 2004.

[25] J. Sloboda, *The Musical Mind: The Cognitive Psychology of Music*. New York: Oxford University Press, 1985.

[26] C. Drake and C. Palmer, "Skill acquisition in music performance: Relations between planning and temporal control," *Cognition*, vol. 74, no. 1, pp. 1–32, 2000.

[27] C. Palmer, "Music performance," *Annual Review of Psychology*, vol. 48, pp. 115–138, 1997.

[28] O. Beringer and T. F. Dunhill, *Manual of Scales, Arpeggios, and Broken Chords for Pianoforte*. London: The Associated Board of the Royal Schools of Music, 1989.

[29] H. Guan, Z. Yan, and T. Hsu, "Automatic piano fingerings estimation using recurrent neural networks," in *Proceedings of the 2nd Nordic Sound and Music Computing Conference*, November 2021.

[30] X. Guan, H. Zhao, and Q. Li, "Estimation of playable piano fingering by pitch-difference fingering match model," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2022, pp. 1–13, 2022.

[31] R. M. Sherman and R. B. Sherman, "Scales and arpeggios," in *The Aristocats*. The Walt Disney Company, 1970.

[32] D. A. Randolph, J. Badgerow, C. Raphael, and B. Di Eugenio, "Pydactyl: A Python framework for piano fingering," in *Extended Abstracts for the Late-Breaking Demo Session of the 19th International Society for Music Information Retrieval Conference*, Paris, 2018.

[33] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, Utrecht, Netherlands, 2010, pp. 637–642.

[34] O. M. Bjørndalen, "Mido," https://pyserial.readthedocs.io, 2020, Accessed: 2023-03-22.

[35] A. Allam and M. Krauthammer, "PySeqLab: An open source python package for sequence labeling and segmentation," *Bioinformatics*, vol. 33, pp. 3497–3499, 2017.