Pydactyl: A Python Framework for Piano Fingering

David A. Randolph[1], **Justin Badgerow**[2], **Christopher Raphael**[3] **and Barbara Di Eugenio**[1]

[1] Department of Computer Science, University of Illinois at Chicago
[2] Division of Music, Elizabethtown College
[3] Department of Computer Science, Indiana University Bloomington
drando2@uic.edu

1. Overview

Object-oriented Python framework for rapid development of fingering models. Built on music21 [2] and standardizes

- Problem decomposition
- Data formats
- Fingering segmentation and re-combination
- Evaluation methods

Reference implementations of 3 published models [6, 4, 3].

2. Implementation

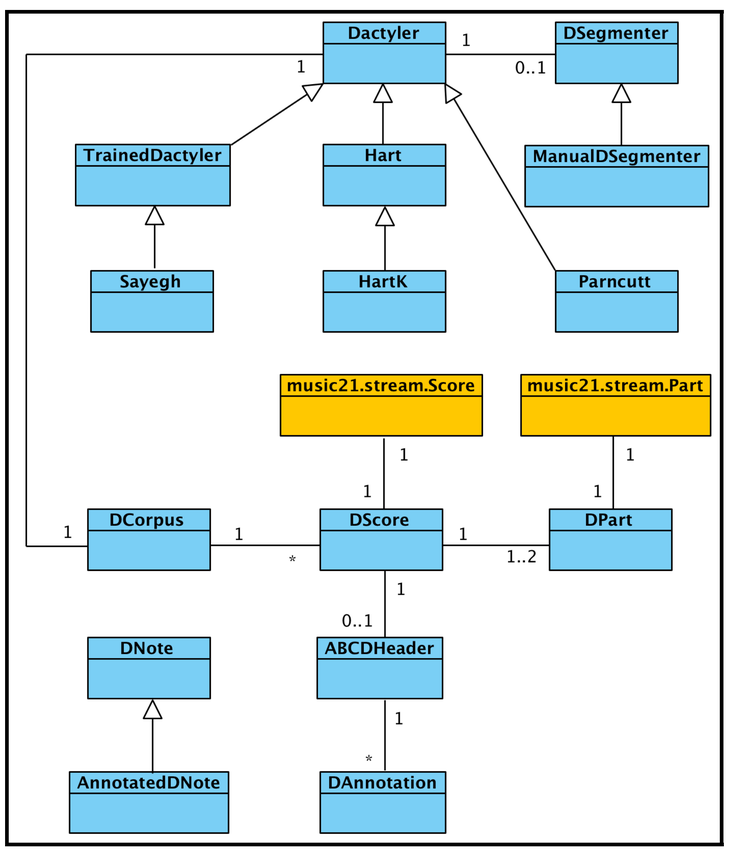
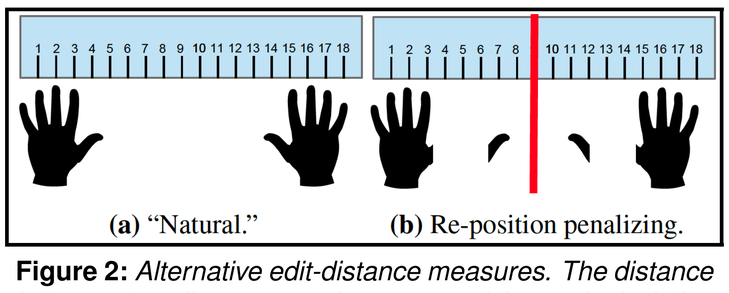


Figure 1: *Pydactyl UML class diagram.*

3. Evaluation Methods

Supports set of evaluation metrics to compare model output to gold-standard corpora:

- Pivot (mis-)alignment measure
- Standard Hamming distance
- A more "natural" weighted edit distance[1]
- An edit distance detecting hand re-positioning and penalizing accordingly[2]



(a) "Natural." **(b)** Re-position penalizing.

Figure 2: *Alternative edit-distance measures. The distance between two fingers on ruler assessed for each deviation detected when comparing one suggested fingering to another.* *Credit: Hand by Baranovskiy [1].*

Also, a novel "re-entry cost" measure, requiring multiple advice generations to assess a model:

1. Check the alignment of two fingering sequences.
2. At first deviation from gold standard,
   (a) Impose an edit-distance cost;
   (b) Re-generate advice on sub-sequence of notes from point of deviation, constraining first fingering to match gold standard;
   (c) Go to step 1 until all notes are processed.

[1]The "natural" measure penalizes each individual finger deviation by the absolute difference in the expected and actual fingering numbers.
[2]This "re-position penalizing" measure infers hand re-positioning from deviations involving the thumb and penalizes such differences more severely.

All models in framework must support constraining first and last fingerings for any segment.

4. Example Use Case

1. Experts enter data using updated, purpose-built web interface, abcDE [5].
   - Fingering specification
   - Phrase demarcation
2. Experts generate standardized abcDF (version 6) content.
3. Researchers load (training/evaluation) abcDF data.
4. Researchers use Pydactyl for new fingering models:
   - Phrase segmentation
   - Segmented fingering recombination
   - Full underlying music21 functionality
   - Predefined intrinsic evaluation measures

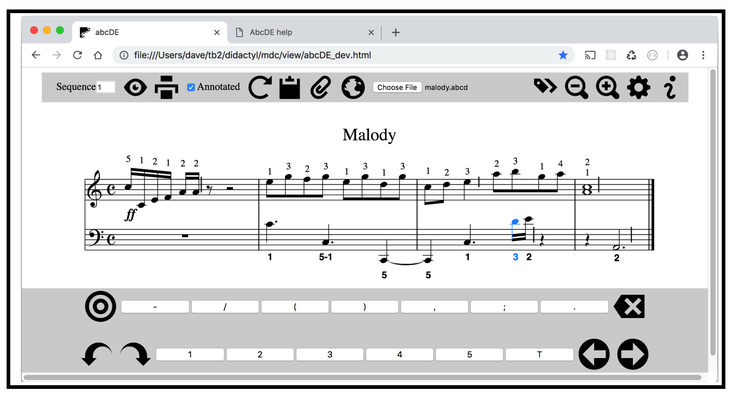


Figure 3: *Screen capture of updated abcDE in action.*

5. Revised abcD File Format

abcDE's native file format, "abcD," extends abc to augment an abc score with

- Fiingering details
- Data provenance
- Phrase segmentation (new!)

```
% abcDidactyl v6
% abcD fingering 1: 512122.13231313123;231412.@15-15
51;32.2.
% Authority: Joe Ivory (1847)
% Transcriber: David Bartleby
% Transcription date: 2018-08-21 12:49:02
% Joe Ivory is a mythical pianist.
% This is an example.
% abcDidactyl END
X:1
X:1
T:Malody
M:C
K:C
V:1 treble
!ff!c/C/E/F/ A/A/zz4|egfg egdg|cD'e2 abga|\
[Ac]8|]
V:2 bass
z8|C3C,3 C,,2-|C,,2 C,3 D/E/ z2|z2 A,,6|]
```

Listing 1*: abcD file example.*

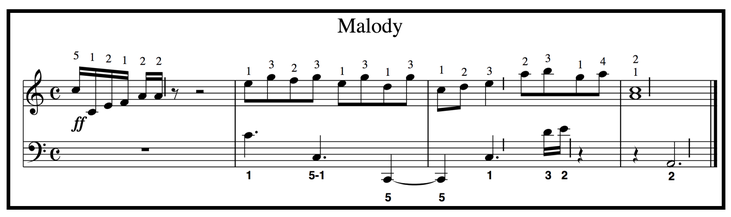


Figure 4: *Rendered abcD file example with horizontal-line phrase markings, as specified in [7, §4.14].*

6. Augmented abcDF Language

EBNF grammar, now including phrase demarcation as in Listing 2, allows parsing of abcD by abcDE and Pydactyl.

```
sequence := staff ("@" staff)?
staff := line "&" staff | line | ""
line := (score_fingering)*

score_fingering := ornamental ("/" ornamental)?
(segmenter)?
| pedaled_fingering ("/" pedaled_fingering)?
(segmenter)?
| pedaling ("/" pedaling)? (segmenter)?

ornamental := "(" (pedaled_fingering)+ ")"

pedaling := (soft)? "x" (damper)?
pedaled_fingering := (soft)? fingering (damper)?

fingering := finger ("-" finger)?
finger := (hand)? digit

segmenter := "," | ";" | "."
damper := "_" | "^"
soft := "p" | "f"
hand := "<" | ">"
digit := "1" | "2" | "3" | "4" | "5"
```

Listing 2*: Extended Bachus-Naur Form (EBNF) grammar defining the augmented abcDF format.*

7. Example Usage

```
model = Parncutt(segmenter=ManualDSegmenter(),
                 segment_combiner="cost")
d_corpus = DCorpus(paths=["/tmp/malody.abcd"])
model.load_corpus(d_corpus=d_corpus)
advice = model.advise()
# Gold-standard embedded in input file.
hamming_dists = model.evaluate_strike_distance()
```

Listing 3*: Pydactyl usage example.*

8. Deployment

Puydactyl source hosted at `https://github.com/dvdrndlph/pydactyl`, To install, type

```
pip3 install pydactyl
```

Example abcDE instance at `http://dvdrndlph.github.io/didactyl/abcde/view/abcDE_dev`.

All project code is open source and released under the terms of the MIT license.

9. Acknowledgments

This work has been supported by a Provost's Award from the University of Illinois at Chicago and a Faculty Grant from Elizabethtown College. Overdue thanks to Zoe Randolph.

References

[1] Dmitry Baranovskiy. Hand. `https://thenounproject.com/DmitryBaranovskiy`. Copyright information: CC-BY 3.0 license. Accessed: 2017-04-25.

[2] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*, pages 637–642, 2010.

[3] Melanie Hart, Robert Bosch, and Elbert Tsai. Finding Optimal Piano Fingerings. *The UMAP Journal*, 21(2):167–177, 2000.

[4] Richard Parncutt, John A. Sloboda, Eric F. Clarke, Matti Raekallio, and Peter Desain. An ergonomic model of keyboard fingering for melodic fragments. *Music Perception*, 14(4):341–382, 1997.

[5] David A. Randolph and Barbara Di Eugenio. Easy as abcDE: Piano fingering transcription online. In *Extended Abstracts for the Late-Breaking Demo Session of the 17th ISMIR Conference*, 2016.

[6] Samir I. Sayegh. Fingering for string instruments with the optimum path paradigm. *Computer Music Journal*, 13(3):76–84, 1989.

[7] Chris Walshaw. The abc Music Standard 2.1. `http://abcnotation.com/wiki/abc:standard:v2.1`, 2011. Accessed: 2016-06-28.