Musical Composer Identification and Generation

Mildness Onyekwere & Shaun Thornton

Contents

Motivations

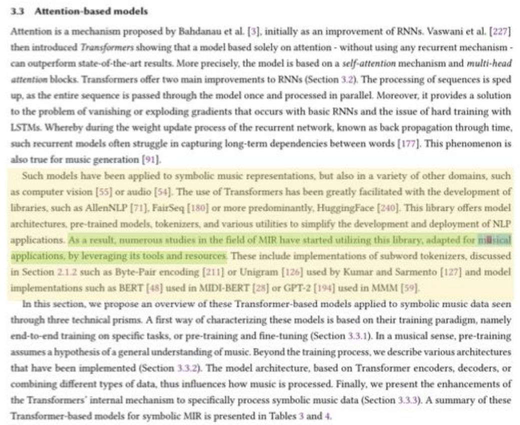
When originally brainstorming a direction for this project, we both wanted music to be at its center.

- Music is, in a sense, it’s own language that parallels the concepts of human language quite well.

- Music is something that can be processed auditorily, but it can also be seen in written formats.

Both of these qualities lend themselves well to Natural Language Processing

Motivations

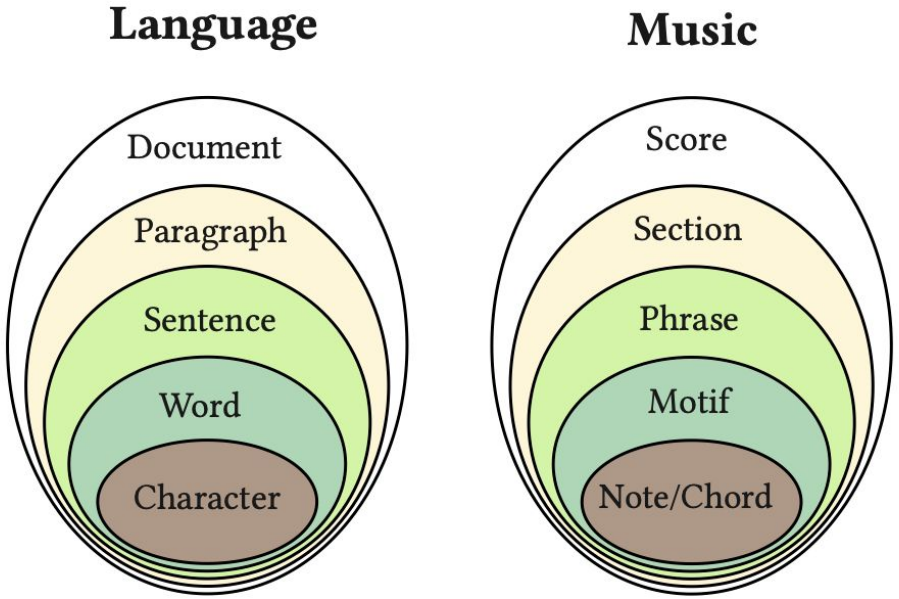


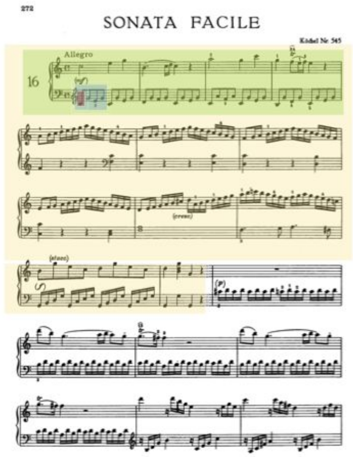
3.3 Attention-based models

Attention is a mechanism proposed by Bahdanau et al. [3], initially as an improvement of RNNs. Vaswani et al. [227] then introduced *Transformers* showing that a model based solely on attention - without using any recurrent mechanism - can outperform state-of-the-art results. More precisely, the model is based on a *self-attention* mechanism and *multi-head attention* blocks. Transformers offer two main improvements to RNNs (Section 3.2). The processing of sequences is sped up, as the entire sequence is passed through the model once and processed in parallel. Moreover, it provides a solution to the problem of vanishing or exploding gradients that occurs with basic RNNs and the issue of hard training with LSTMs. Whereby during the weight update process of the recurrent network, known as back propagation through time, such recurrent models often struggle in capturing long-term dependencies between words [177]. This phenomenon is also true for music generation [91].

Such models have been applied to symbolic music representations, but also in a variety of other domains, such as computer vision [55] or audio [54]. The use of Transformers has been greatly facilitated with the development of libraries, such as AllenNLP [71], FairSeq [180] or more predominantly, HuggingFace [240]. This library offers model architectures, pre-trained models, tokenizers, and various utilities to simplify the development and deployment of NLP applications. As a result, numerous studies in the field of MIR have started utilizing this library, adapted for musical applications, by leveraging its tools and resources. These include implementations of subword tokenizers, discussed in Section 2.1.2 such as Byte-Pair encoding [211] or Unigram [126] used by Kumar and Sarmento [127] and model implementations such as BERT [48] used in MIDI-BERT [28] or GPT-2 [194] used in MMM [59].

In this section, we propose an overview of these Transformer-based models applied to symbolic music data seen through three technical prisms. A first way of characterizing these models is based on their training paradigm, namely end-to-end training on specific tasks, or pre-training and fine-tuning (Section 3.3.1). In a musical sense, pre-training assumes a hypothesis of a general understanding of music. Beyond the training process, we describe various architectures that have been implemented (Section 3.3.2). The model architecture, based on Transformer encoders, decoders, or combining different types of data, thus influences how music is processed. Finally, we present the enhancements of the Transformers' internal mechanism to specifically process symbolic music data (Section 3.3.3). A summary of these Transformer-based models for symbolic MIR is presented in Tables 3 and 4.





Parallels between musical and human language structures

Hypotheses

Hypothesis #1: If we train a model on musical data using natural language processing techniques, then we expect it to be capable of predicting the artist/composer of a given piece with high accuracy.

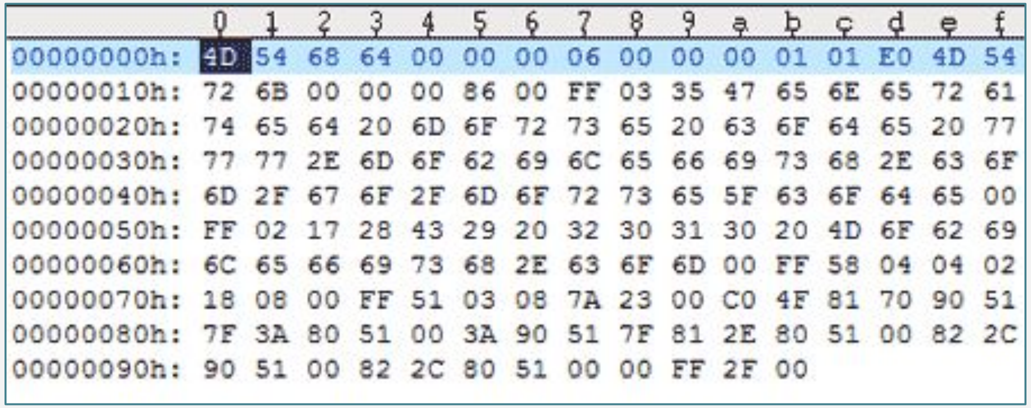
Hypothesis #2: If we create a set of models tailored to individual composers, which uses the first model to evaluate its accuracy, then it should be able to generate novel musical pieces in the style of the desired artist/composer through repeated predictions.

Keywords:
Music Information Retrieval, Feature Extraction, NLP Techniques, Authorship Identification, Content Generation

The Musical Instrument Digital Interface

- Binary format for storing musical data
  - Efficiently stores information, but is not human readable
  - Would not be handled well by traditional NLP techniques

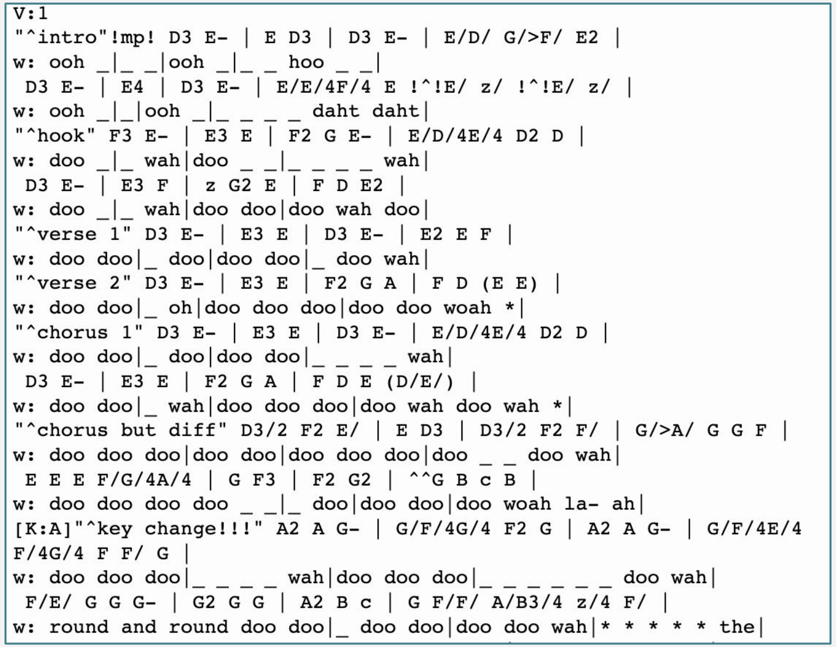
- Our dataset (discussed later) provides all samples in MIDI format
  - The initial input and final output of our entire process will be in MIDI form
  - But we need a more suitable format for model training/predicting



```
             0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 4D 54 68 64 00 00 00 06 00 00 00 01 01 E0 4D 54
00000010h: 72 6B 00 00 00 86 00 FF 03 35 47 65 6E 65 72 61
00000020h: 74 65 64 20 6D 6F 72 73 65 20 63 6F 64 65 20 77
00000030h: 77 77 2E 6D 6F 62 69 6C 65 66 69 73 68 2E 63 6F
00000040h: 6D 2F 67 6F 2F 6D 6F 72 73 65 5F 63 6F 64 65 00
00000050h: FF 02 17 28 43 29 20 32 30 31 30 20 4D 6F 62 69
00000060h: 6C 65 66 69 73 68 2E 63 6F 6D 00 FF 58 04 04 02
00000070h: 18 08 00 FF 51 03 08 7A 23 00 C0 4F 81 70 90 51
00000080h: 7F 3A 80 51 00 3A 90 51 7F 81 2E 80 51 00 82 2C
00000090h: 90 51 00 82 2C 80 51 00 00 FF 2F 00
```

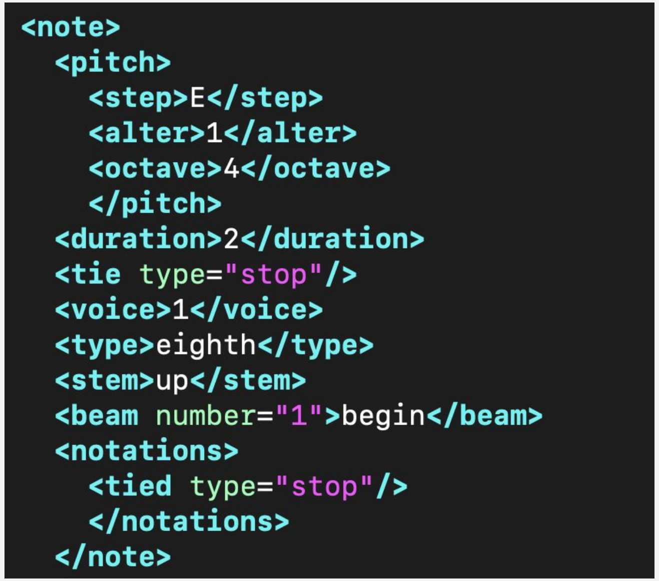
MIDI file in hex editor

ABC Notation vs. MusicXML



```
V:1
"^intro"!mp! D3 E- | E D3 | D3 E- | E/D/ G/>F/ E2 |
w: ooh _|_ _|ooh _|_ _ hoo _ _|
 D3 E- | E4 | D3 E- | E/E/4F/4 E !^!E/ z/ !^!E/ z/ |
w: ooh _|_|ooh _|_ _ _ _ daht daht|
"^hook" F3 E- | E3 E | F2 G E- | E/D/4E/4 D2 D |
w: doo _|_ wah|doo _ _|_ _ _ _ wah|
 D3 E- | E3 F | z G2 E | F D E2 |
w: doo _|_ wah|doo doo|doo wah doo|
"^verse 1" D3 E- | E3 E | D3 E- | E2 E F |
w: doo doo|_ doo|doo doo|_ doo wah|
"^verse 2" D3 E- | E3 E | F2 G A | F D (E E) |
w: doo doo|_ oh|doo doo doo|doo doo woah *|
"^chorus 1" D3 E- | E3 E | D3 E- | E/D/4E/4 D2 D |
w: doo doo|_ doo|doo doo|_ _ _ _ wah|
 D3 E- | E3 E | F2 G A | F D E (D/E/) |
w: doo doo|_ wah|doo doo doo|doo wah doo wah *|
"^chorus but diff" D3/2 F2 E/ | E D3 | D3/2 F2 F/ | G/>A/ G G F |
w: doo doo doo|doo doo|doo doo doo|doo _ _ doo wah|
 E E E F/G/4A/4 | G F3 | F2 G2 | ^^G B c B |
w: doo doo doo doo _ _|_ doo|doo doo|doo woah la- ah|
[K:A]"^key change!!!" A2 A G- | G/F/4G/4 F2 G | A2 A G- | G/F/4E/4
F/4G/4 F F/ G |
w: doo doo doo|_ _ _ _ wah|doo doo doo|_ _ _ _ _ _ doo wah|
 F/E/ G G G- | G2 G G | A2 B c | G F/F/ A/B3/4 z/4 F/ |
w: round and round doo doo|_ doo doo|doo doo wah|* * * * * the|
```

ABC Notation
(Dense, but more english-like than MIDI)



```xml
<note>
  <pitch>
    <step>E</step>
    <alter>1</alter>
    <octave>4</octave>
    </pitch>
  <duration>2</duration>
  <tie type="stop"/>
  <voice>1</voice>
  <type>eighth</type>
  <stem>up</stem>
  <beam number="1">begin</beam>
  <notations>
    <tied type="stop"/>
    </notations>
  </note>
```

MusicXML
(Very readable, but also verbose)
(This entire snippet represents a single note)

Methods and Tools

- **pretty-midi:** Python library for manipulating MIDI files
  - Part of our custom tool that converts MIDI into ABC Notation/MusicXML, *and vice-versa*

- **SentencePiece:** An unsupervised text tokenizer and detokenizer for Neural Network-based text generation systems (Google)
  - Tokenize our musical data, following its conversion to textual form

- **Word2Vec:** Converts words into word embeddings (captures their semantic meaning)
  - Generate embeddings from the tokenized musical data

- **sklearn:** Python machine learning library
  - Perform composer classification (KNeighborsClassifier)

- **Recurrent neural network (RNN):** Good at text generation (ABC notation / MusicXML)
  - Generate a piece through repeated predictions

The MAESTRO Dataset

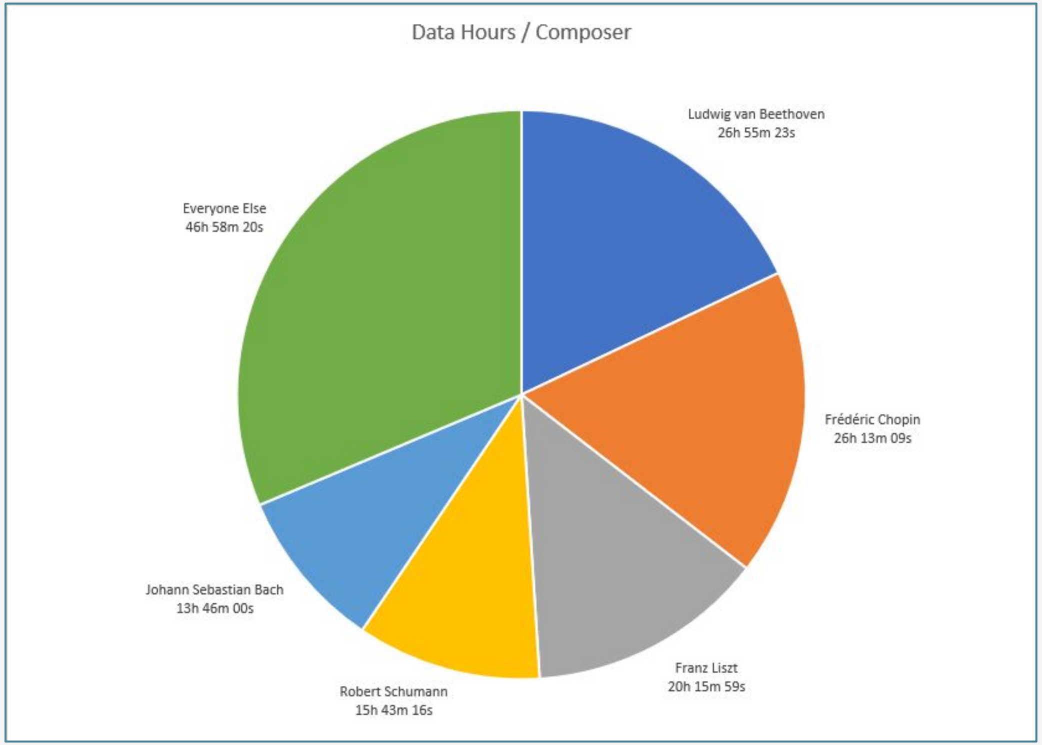
- ~200 hours of highly-accurate MIDI recordings
  - Taken from the Minnesota International Piano-e-Competition
    - ~20 year-long run (2002-2021)
  - Includes metadata for each piece
    - Composer, title, year performed
  - Provides a high quality train/validation/test split
    - Though we have created our own splits instead

- Released as part of a corresponding paper
  - *Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset*
  - Trained models capable of transcribing/composing audio waveforms
    - Used a novel process they called “Wave2Midi2Wave”
  - We do *not* intend to train our models on audio waveforms
    - This paper is *not* directly applicable to out work
    - The dataset compiled for the paper is *very* applicable

The MAESTRO Dataset Analysis

- The top 5 composers account for nearly ¾ of the entire dataset.
  - *Fun fact: There was a 45 minute long Bach entry*

- We want our set of composers to have approximately equal representation for authorship identification
  - Cut the top 5 down to around 13h 45m each
  - If this cut hurts accuracy, we can reduce the number of composers to just 3 (~20h each)



The MAESTRO Dataset Splits

- We have prepared our own set of train/validate/test splits for *both* the five and three most frequently played composers. These will be used for composer identification (first model):
    - Ludwig van Beethoven
    - Frédéric Chopin
    - Franz Liszt
    - Robert Schumann
    - Johann Sebastian Bach

- We have also prepared individual train/validate/test splits for the top three composers. These will be used to train the composer content generation models.
    - Ludwig van Beethoven
    - Frédéric Chopin
    - Franz Liszt

- The first (authorship identification) model will be used to determine the quality of the content generated by the second set of (piece generation) models.
    - i.e. If we produce a piece in the style of Beethoven, we want the first model to identify it as being by Beethoven.

Natural Language Processing Methods for Symbolic Music Generation and Information Retrieval

Dinh-Viet-Toan Le, Louis Bigo, Dorien Herremans, and Mikaela Keller. 2025

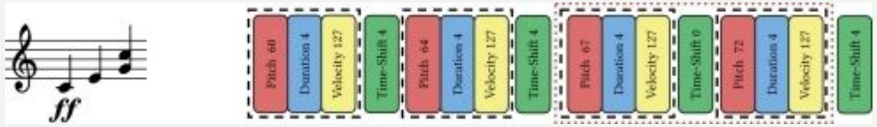
Overview:

- Explores how different NLP methods can be used for symbolic music generation and information retrieval.
- Provides an in-depth overview of the options available, then evaluate various strategies and models to determine where/how each performs best.

Relevance:

- This research is strongly relevant to the second model we’re attempting to create, since its goal will be to generate music symbolically in the style that it is trained in.

Natural Language Processing Methods... *Continued*



Tokenization Strategies

- Time-Sliced vs Event Based
  - Divide music into fixed temporal slices, and represent which notes (or events) happen in each slice.
  - Represent music as a sequence of events (e.g., “note-on”, “note-off”, “time-shift”, “velocity”). This is very analogous to tokenizing text into words.
- Composite vs Elementary
  - Elementary tokens might be very basic events (e.g., a single note-on). Composite tokens combine multiple musical features (pitch, duration, velocity, etc.).

Creating a Model with MIR

- After tokenizing we can create an embeddings layer that can either be static (word2vec) or contextual (transformer embeddings)

Natural Language Processing Methods... *Continued*

Model Strategies

- Recurrent Models:
  - RNNs, LSTMs, GRUs: Traditional sequence models.
  - Can suffer from the issue of vanishing gradient occurring with long sequences, which is often the case in symbolic music
- Transformer Based:
  - *End-to-end training*: Training transformers directly on a generation or retrieval task.
  - *Pre-training + fine-tuning*: Similar to BERT or GPT in NLP, models are pre-trained on large unlabeled corpora of symbolic music, then fine-tuned for downstream tasks.
  - Encoder-Only (BERT like)
    1. Bidirectional models have led to symbolic music adaptations of BERT such as MuseBERT, MusicBERT, MidiBERT-Piano, etc..
  - Decoder-Only (GPT like)
    1. By comparing multiple decoder-only architectures, such pre-trained decoder-only models appear to perform better in piano generation
- Music Specific Adaptations
  - Bar-level masking: In pre-training, instead of masking individual tokens (notes), entire bars or groups of features are masked to prevent trivial information leakage, akin to BERT's masked language modeling but adapted for musical structure.
  - Positional encoding / attention tweaks: Because music has special structure (bars, beats, time-signature, tempo), transformers used for symbolic music often modify the positional encoding or attention to account for musical features
  - Domain-knowledge embeddings: Some models integrate musical knowledge (e.g., pitch intervals, relative timing) into the embedding space, so the model better captures musical relationships.

NLP-based music processing for composer classification

Deepaisarn, S., Chokphantavee, S., Chokphantavee, S. et al. 2023

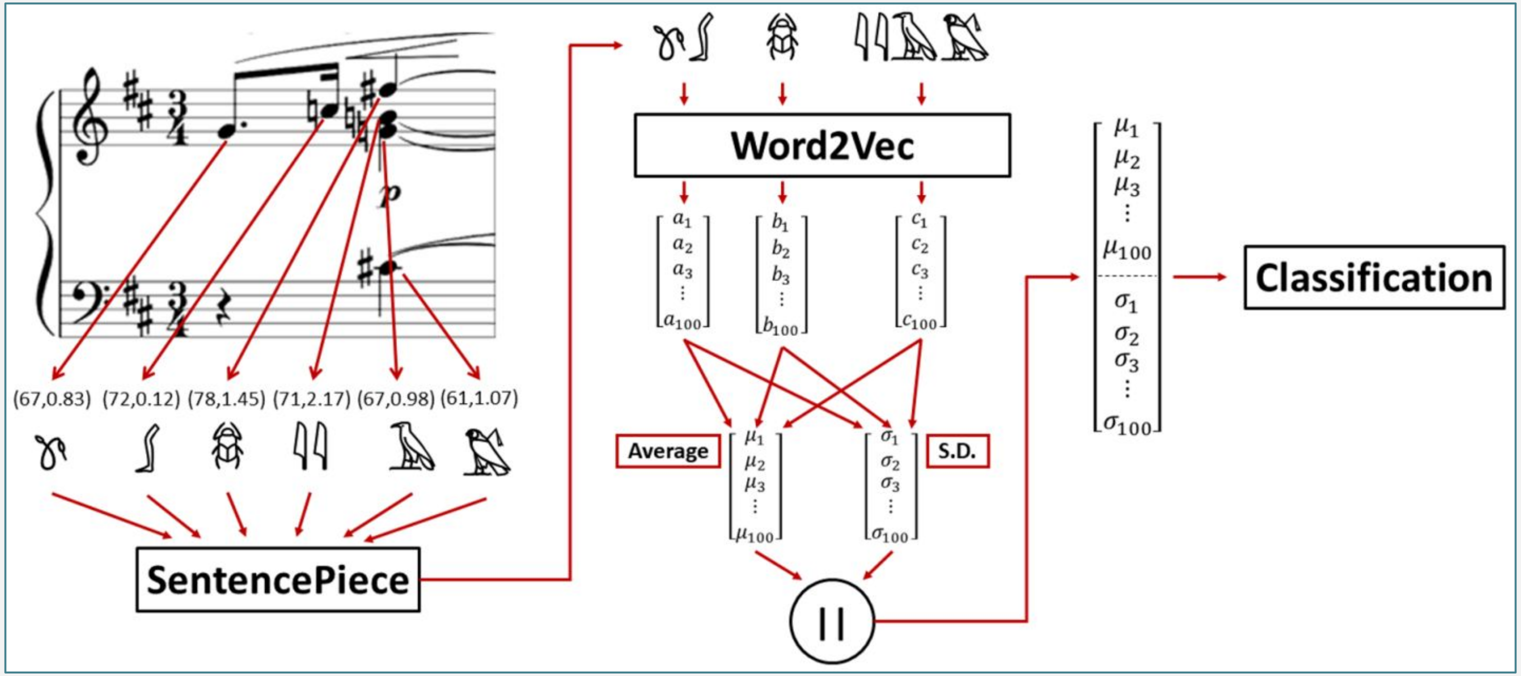
Overview:

- Performed composer classification with the MAESTRO dataset using SentencePiece and Word2vec.
  - Their data splits were suboptimal, choosing to select the top artists according to how many times they were played rather than their total play time
  - Data splits were also highly unbalanced, which the paper often cited as a hindrance to the model's performance, yet they never attempted to fix the issue

Relevance:

- The process described in this paper highly resembles what we're attempting to achieve with our first model
  - It is crucial that we get this first model right, as it will be used to judge the accuracy of the composer-specific generation models
  - There are clear improvements that can be made are part of our process

NLP-Based Music Processing for Composer Classification... *Continued*



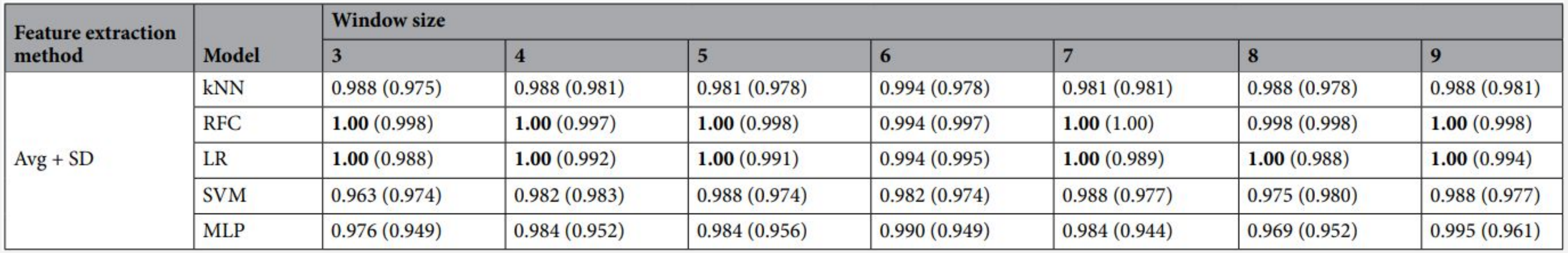
NLP-Based Music Processing for Composer Classification… *Continued*

Tested (5) different classification models:

- K-nearest neighbors (kNN)
- Random forest classifier (RFC)
- Logistic regression (LR)
- Support vector machines (SVM)
- Multilayer perceptron (MLP)

All five models performed *exceedingly* well, holding 96+% accuracy across the board, regardless of window size.

kNNs are the simplest model / the one we understand best, which is why we selected it as our preferred option.



Feature extraction method	Model	Window size						
		3	4	5	6	7	8	9
Avg + SD	kNN	0.988 (0.975)	0.988 (0.981)	0.981 (0.978)	0.994 (0.978)	0.981 (0.981)	0.988 (0.978)	0.988 (0.981)
	RFC	**1.00** (0.998)	**1.00** (0.997)	**1.00** (0.998)	0.994 (0.997)	**1.00** (1.00)	0.998 (0.998)	**1.00** (0.998)
	LR	**1.00** (0.988)	**1.00** (0.992)	**1.00** (0.991)	0.994 (0.995)	**1.00** (0.989)	**1.00** (0.988)	**1.00** (0.994)
	SVM	0.963 (0.974)	0.982 (0.983)	0.988 (0.974)	0.982 (0.974)	0.988 (0.977)	0.975 (0.980)	0.988 (0.977)
	MLP	0.976 (0.949)	0.984 (0.952)	0.984 (0.956)	0.990 (0.949)	0.984 (0.944)	0.969 (0.952)	0.995 (0.961)

F1 scores on test and validation dataset (parenthesis)

Risks (and *MIDI*-gations)

- The accuracy of the composer identification model is *crucial*
  - Error/poor accuracy within this first model could compound into far worse accuracy issues in the subsequent composer-specific models
  - We need to ensure that the composer identification model is extremely accurate. Our contingency data splits should help here.

- RNNs often fail to stay on-track when generating longer content
  - We're currently unsure as to how/when this issue would manifest, if at all
  - This wouldn't only affect the quality of the generation, but could also break it completely since we're generating structured data (ABC notation / MusicXML)
  - Hoping that generating only short pieces (~30s) won't pose a problem.

- Our generation strategy may not work *at all*
  - Transformer-based solutions exist, such as MuseBERT / MusicBERTMidi. We could always transition over to these if needed, but we wan't to always keep NLP at the core of our process

Questions?