Recognizing Classical Composers Using High-Level Musical Features

Achim Koh
MA in Liberal Studies
The Graduate Center, CUNY
New York, New York 10016
Email: akoh@gradcenter.cuny.edu

***Abstract*—This paper addresses the automatic classification of classical music scores according to their composer, by applying methods common in text classification. Event durations and sequences of approximated chords are extracted from the dataset. Tf-idf values are computed for durations and chord n-grams, then used to build five composer classifier models: support vector machine, logistic regression, k-nearest neighbors, Naive Bayes and multilayer perceptron. The support vector machine is shown to work best in both two-class and multiclass settings.**

***Keywords*—music classification, high-level musical feature, chord n-grams, duration, machine learning, music21, scikit-learn.**

I. Introduction

Identifying the composer by listening to a piece of music is a task often associated with human knowledge of music. An entrance exam for the composition program of a conservatory will typically require the candidate to listen to a piece of music without knowing its composer or title, then to identify the stylistic elements as well as historical context pertinent to the composer of that piece. In other words, recognizing composers usually requires the combined knowledge of music theory and history.

This research tackles the previously mentioned task using machine learning methods. Without heavily relying on expert knowledge, this data-driven approach seeks to build accurate predictive classification models using features collected from labeled data. Specifically, it focuses on the usage of chords by each composer and applies methods common in information retrieval and text mining, such as n-grams and tf-idf values, to the classification of musical scores. In addition, the durations of notes, chords and rests are also considered as features.

While previous work on computational composer classification goes at least as far back as 1958 [1], data-driven work on music classification based on composer has been rather limited to the last decade [2]. In this experiment I follow the viewpoint of [3] who regard that sequences of chords can be treated like word sequences. Under this viewpoint, text classification techniques are applicable to musical features. There exists a long history of parallel between music and language, such as the medieval emphasis on formal syntax [4], the more recent turn to linguistics including Chomsky-influenced musical grammar models [5] and work pointing to the neuroscientific resemblance of the two activities [6]. This parallel and potential similarity is explored through the usage of text classification methods on music scores.

Especially, this research is interested in tf-idf and n-grams, representation techniques commonly used for representation in information retrieval and text mining [7]. N-grams have been explored in musical machine learning by [8] and [9] among others. More specifically, the use of chord n-grams in order to represent and analyze music has been explored by studies such as [10], [11] and [12]. In addition, the experiment follows [13]'s recommendation to include note duration features; [14] have also explored the potential of durations as useful features in music classification tasks. A more comprehensive overview of historical work done on automatic composer classification, as well as the field of music information retrieval in general, is provided by [2]. The experiment examines five composer classifier models, support vector machine (SVM), logistic regression (LR), k-nearest neighbors (kNN), Naive Bayes (NB) and multilayer perceptron (MLP), and yields good classification results, especially with the SVM.

II. The Features

A. Data Selection and Preprocessing

The database[1] used in this experiment includes 17,382 MIDI files by over 900 composers pre-labeled with the composer, among which 4,082 files by 5 composers were selected: J. S. Bach (BA), Beethoven (BE), Debussy (DE), Scarlatti (SC) and Victoria (VI). These composers were selected among the top frequent composers in the database in order to have enough pieces per composer to enable cross-validation. They were also selected so that each combination of two composers has at least one difference in either nationality or era. These rather broad categories are commonly used when describing a composer's style. While the relation of these factors and classification results is not absolute and also beyond the scope of this experiment, they are helpful in ensuring that the selected composers will have at least some stylistic differences. Victoria (1548-1611) was a Spanish composer. Bach (1685-1750) and Scarlatti (1685-1757) are both categorized as Baroque composers, but Bach was German and Scarlatti was an Italian who mainly served the Portuguese and Spanish royal families. Beethoven (1770-1827) was German and is considered a Classical and Romantic figure. Debussy (1862-1918) was French and employed non-traditional scales, making him perhaps the most distinct composer in this dataset.

Preprocessing and feature extraction was done using music21. Music21 is a Python library that provides a toolkit

[1] http://www.haralick.org/ML/k_collection.zip

TABLE I. DATASET

Composer	# MIDI	# Preprocessed
J. S. Bach (BA)	2268	2245
Beethoven (BE)	684	667
Debussy (DE)	199	199
Scarlatti (SC)	598	598
Victoria (VI)	333	333
Total	4082	4042

TABLE II. PITCH CLASSES

Note	C	D♭	D	E♭	E	F	G♭	G	A♭	A	B♭	B
Pitch Class	0	1	2	3	4	5	6	7	8	9	10	11

the reading and writing as well as analyzing musical scores [15]. The preprocessing consisted of converting MIDI files to MusicXML files in order to avoid occasional errors that occurred when loading a MIDI file using the music21 package, and removing files smaller than a given size (2kB) that contain no notes. After preprocessing, most of the originally selected files (4,042) were used. Table I shows an overview of the selected pieces.

B. A Few Remarks on Musical Terms

This research considers the note as the most basic element in a musical score. A note indicates the presence of a sound in a specific time position, at a specific pitch, and for a specific duration. In the context of traditional Western music, the pitch of a note can have one of 12 vertically different classes, each a semitone away from the previous one. Semitones are defined so that the 12th semitone away from a note will have a frequency double (when moving up) or half (when moving down) that of the starting note. This interval of 12 semitones, or between a frequency and its half, is called an octave. Notes that are one or several octaves away from each other are said to have the same pitch class. Two notes with the same pitch class, even when they are in different octaves, are treated as equivalent within the scope of this experiment. Here we define pitch classes like Fig II, so that a C note will have a pitch class of 0. Note that the pitch classes constitute a circular array, i.e. the shortest distance between C and B is not 11 but 1.

A chord is a set of multiple different notes played simultaneously. Chords are defined in terms of the root note, along with different musical properties that relate with the combinations of notes. The root note corresponds to the fundamental frequency of a chord. Since notes with the same pitch class are equivalent, the same chord can be built using notes that are in higher or lower octaves; therefore, the name "root" does not necessarily mean that it is the lowest note. Rather, it can be thought of as the reference point from which the relations among notes are defined. One important property that a chord can have is called quality: the quality is determined by whether the chord contains a note three semitones above the root, or four semitones above it. It is rare that one chord would contain both; the former is called a minor chord, and the latter a major chord. Chords also have other properties than their qualities, and some chord types often used in Western music are defined in Table III.

Note that the pitch classes in Table III show chords with a C root. Since a chord type is defined based on its root note, a single type of chord exists in 12 possible positions in terms of pitch. If we moved all the notes in a chord upward by a semitone each, the result would be the same type of chord but with a different root. For example, a D Major chord would be in this case {2,6,9}. With 12 possible roots for each one of the 16 chord types, we have a total of 192 pre-defined known chords. In this experiment, we assume that any given chord either corresponds to or can be approximated to one of these 192. Moving all notes by a given interval like previously mentioned is called transposition.

TABLE III. PRE-DEFINED CHORDS (ROOT: C)

Chord	Pitch Class Set
Major	{0,4,7}
Minor	{0,3,7}
Suspended	{0,5,7}
Augmented	{0,4,8}
Diminished	{0,3,6}
Major Sixth	{0,4,7,9}
Minor Sixth	{0,3,7,9}
Dominant Seventh	{0,4,7,10}
Major Seventh	{0,4,7,11}
Minor Seventh	{0,3,7,10}
Half Diminished Seventh	{0,3,6,10}
Diminished Seventh	{0,3,6,9}
Major Ninth	{0,2,4,7,11}
Dominant Ninth	{0,2,4,7,10}
Dominant Minor Ninth	{0,1,4,7,10}
Minor Ninth	{0,2,3,7,10}

A key refers to a group of pitches upon which a piece of music (or part of it) is composed. The group is defined in terms of the tonic which is the reference note, in a similar manner to the root of a chord; and the mode which can be major or minor. While there exist various kinds of modes, those differences are beyond the scope of this paper; instead, I will state, with the risk of vexing rigorous musical theorists, that a major key in a specific tonic operates within the same group of pitches as the minor key in the tonic three semitones below. This brings us to one final remark: when a piece is transposed, i.e. all of the notes constituting the piece are moved by a fixed interval, the resulting notes and chords will maintain the same relations among them; but they will exist in a different key, and chords will have different roots.

C. Chords, Tf-idf, and N-grams

One way to think of a piece of traditional Western music is as a sequence of chords. Western music heavily emphasizes harmony, and the sequential usage of chords is a key element in establishing harmonic structure and attaining affective characteristics [3]. Chords are also important in the perception of music, and serve as background for melody [10]. In other words, a chord is an audio-temporal block that the composer lays down, serving as platform on which the musical piece progresses.

This research assumes that the usage of chords, i.e. the frequency of chords and sequences of chords are an indicator of a composer's style. The strict emphasis on harmony in Western classical music implies that given a specific key, there are only so many chords that a composer can employ in a piece. This constraint given by the cultural convention is, however, not absolute; there is room for originality, which

TABLE IV. CHORD APPEARANCES IN DATASET

	BA	BE	DE	SC	VI
mean	4282.30	3653.15	478.84	965.93	780.68
std	8649.46	5626.85	532.83	1406.65	1532.89
min	4	17	5	1	1
25%	365.5	820	146.75	165	19.75
50%	1400	1837	331	446	191.50
75%	4667.5	4581	569.25	1210	814
max	83691	52747	3002	9483	9265

takes the form of exceptions from statistical expectation [4]. Based on this constraint, one can imagine that a composer's stylistic choice will involve determining the relative frequency of specific chords, whether intentionally or not. The classical emphasis on harmony also means that given a chord, the possible options for the next chords will be limited. Which chord sequences a composer tends to resort to, and whether the composer employs unusual chord sequences, are also style indicators.

Each piece is converted into a sequence of chords, then the training set is used to build a term frequency-inverse document frequency (tf-idf) matrix. The tf-idf values for each piece in the test set are computed separately based on this matrix. Tf-idf is a statistic which combines both the importance of a feature to a specific sample (tf) and how much information that feature provides with regards to the entire dataset. (idf) In this experiment, tf is computed as follows:

$$\text{tf}(t, d) = 1 + \log f_t$$

where f_t is raw count of term t. Idf is computed as follows:

$$\text{idf}(t) = \log \frac{n}{\text{df}(d, t)} + 1$$

where n is the total number of documents and $\text{df}(d, t)$ is the number of documents that contain term t. Then tf-idf is calculated as [16]:

$$\text{tfidf}(t, d) = \text{tf}(t) \cdot \text{idf}(t, d)$$

The rationale of using tf-idf values is to avoid giving too much weight to terms that occur frequently but carry little information. Given the limited number (192) of chords that we have defined, the music pieces will contain some chords that are so common that their frequency need to be re-weighted in order to be useful. Table IV shows basic chord usage statistics by composer in our dataset, where one can verify big discrepancies between frequently used chords and seldomly used ones; table VI shows the accuracies of classifiers in our task when using respectively tf-idf values, binary occurrences (regardless of how many times a chord appears in a piece, all non-zero counts are converted to 1) and integer counts. Tf-idf values provide better results overall.

Using features such as tf-idf values of single chords results in the loss of contextual information, i.e. the order of chords will be lost. In order to preserve information about the sequences of chords used by composers, n-grams of adjacent chords will also be considered as features. Different values of n are compared; a combination of unigrams and bigrams produce the best results in the tested classification models.

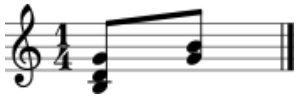

Fig. 1. Beat segment resulting in unique pitch class set {2,7,11}

D. Durations of notes, chords and rests

A preliminary experiment VIII showed that when relying solely on chord n-grams, classifiers performed worse in certain composer pairs, especially when distinguishing BE v. SC. A manual look into the often-misclassified pieces indicated, potentially, different tendencies of note durations by the two composers: in the misclassified pieces, Beethoven tends to use longer notes whereas Scarlatti employs much shorter notes, giving the latter a faster impression. Durations of individual notes, as well as of chords and rests were considered as features in order to improve this performance discrepancy.

Duration refers to the length of time a musical event is sounded. A common way to express durations is by using note values such as whole note, half note, quarter note, eighth note and sixteenth note; the fraction expressed in the name indicates the relative length compared to the beat. The beat is the temporal unit of a piece, defined by the composer in the form of time signature, and often but not always corresponds to a quarter note. By using both chord and duration feature sets, performances of all classifiers except kNN were improved. (Table IX)

E. Feature Extraction

According [2], features used to represent music files can be roughly grouped into low-level features extracted from audio signals (e.g. WAV, mp3), high-level features extracted from structured files (e.g. MIDI, MusicXML), and metadata. This experiment is constrained to work with MIDI files as starting point, which makes the use of high-level features a reasonable choice. Structured music files contain high-level (symbolic) representations of musical events found in a musical score, including but not limited to the presence and duration of notes and rests, key and time signature, and instrumentation.

In this experiment, I choose to use an approximation of each piece's usage of chords as feature set. As previously stated, the assumption is that the usage of chords is an indicator of a composer's style. Because there are several possible ways to interpret a set of notes into a chord, an accurate chord analysis of a piece requires rigorous knowledge of music theory, the analyzed piece, and the composer. Instead of a precise chord analysis, this experiment relies on a model that approximates chords found in a piece to pre-defined chords in Table III.

Each piece is divided into segments of one beat. All notes that are sounding in each beat segment are combined into one single chord, regardless of each note's starting point, end point and duration. This chord is then represented as two sets of pitch classes s_c and s_u, where s_c is count-preserving and s_u consists of unique values. For example, the beat segments illustrated in Fig. 1 will result in the count-preserving set {11,2,7,7,11} and unique set {2,7,11}. If the segment has no note in it, we call it a rest and treat it as a separate chord.

The sets are then transposed from the piece's original key into the C major key, or the A minor key. As implied above, I am here assuming that the each of the 12 major keys are equivalent to one of the 12 minor keys; once each major key is transposed to C, or minor key to A, the pieces are considered to be in the same reference coordinates. The goal here is to see how a composer employs chords, not in terms of absolute pitch, but relatively. This means that I disregard any information pertinent to the composer's choice of key, which is indeed musically significant; however, for the scope of this paper the transposed chords will do. Also, this transposition is done only once for every chord in the piece, which makes it impossible to consider mid-piece key change. However, the unconventional chords registered due to not accounting for key changes might be informative on their own. The transposed pitch class set is compared with 192 predefined known chords that are common in Western classical music. The closest known chord is used as the corresponding beat segment's representation; the piece is represented as a series of thus found known chords. Along with the possible rest, this gives us 193 features.

The distance between pitch class sets c_1 and c_2 is computed as follows. By moving the pitch classes up or down within c_1, adding pitch classes to or removing them from it, one can make it equal to c_2. The cost of moving a note up or down by 1 semitone is 1, and the cost of adding or removing a specific note is 3. The sum of the costs is the distance. For example, the distance between {2,5,7} and {2,4,8,10} is computed as 5. Given an input chord represented as s_c and s_u, the distances between s_u and each of 192 known chords are calculated. If the root of the known chord happens to be the most common pitch in s_c, I assume that it has a higher chance to be the input chord's root too and subtract 1 from the current distance. The chord with the minimum distance is selected as representation.

Each chord or rest's id is used to build a sequence of strings, such as: [-1 -1 -1 -1 17 17 17 17 111 58 142 135 135 135 28 17 . . . 16 16 16 16]. This sequence is used as input for scikit-learn's TfidfVectorizer module which outputs a tf-idf matrix used to train the classifiers, and the CountVectorizer module which can output a matrix of either binary occurrences or integer counts. Table V shows the number of total features by the size of n. A preliminary experiment on chord n-grams using accuracy as performance metric shows that a combination of unigrams and bigrams using tf-idf values yield the best results in terms of accuracy (SVM), as shown in Table VI. Values of $n \geq 5$ produced much worse results and therefore were excluded from the experiment. The following experiments all use tf-idf values of both unigrams and bigrams as feature.

Vectorizing binary occurrences instead of tf-idf values resulted in an accuracy drop for SVM, the best classifier in this experiment. While a test run of 100 iterations resulted in 96.83% mean accuracy (standard deviation (SD): 0.11) for tf-idf, it gave 94.61% (SD: 0.18) for binary occurrences, making SVM the second best classifier behind MLP. In contrast, NB and MLP sometimes benefited from it, especially when $n = 4$. The kNN was turned essentially useless when $n \geq 3$, predicting most pieces as BE.

Using integer counts had overall a similar effect including a larger accuracy drop for SVM, bringing its rank to 3rd best, and additionally affected LR in a largely negative way. Such themes as the smaller drop in accuracy in the case of MLP, the difference in kNN, NB and MLP's behavior with higher n-grams, as well as the big drop in LR's accuracy when vectorizing integer counts, could be explored by future research.

TABLE V. UNIQUE CHORD N-GRAMS IN DATASET

n	BA	BE	DE	SC	VI	Total
1	179	173	172	173	146	182
2	15783	16657	10797	10878	4204	21648
3	145459	132964	38978	57871	22597	274026
4	341141	265577	55635	103655	50087	712690

TABLE VI. ACCURACY (%) OF CLASSIFIERS ON CHORD N-GRAMS (ACCORDING TO FEATURES, BEST IN BOLD)

(Tf-idf)

n	SVM	LR	kNN	NB	MLP
1	93.05	89.09	90.35	83.28	90.75
2	96.96	**94.78**	90.2	88.35	94.32
3	94.36	90.25	88.32	88.25	90.65
4	90.3	83.75	84.76	90.4	83.33
[1,2]	**97.35**	94.63	**91.79**	88.84	**94.98**
[3,4]	93.42	88.1	87.41	89.98	88.17
[1,2,3,4]	96.39	92.21	91.49	**90.7**	90.57

(Binary)

n	SVM	LR	kNN	NB	MLP
1	89.07	85.21	**84.76**	80.26	86.99
2	93.07	88.03	47.3	87.58	95.35
3	87.58	77.41	26.18	86.57	92.31
4	81.1	72.14	23.87	**89.49**	85.77
[1,2]	**95.13**	**90.28**	60.39	87.98	**95.65**
[3,4]	86.89	72.29	24.69	88.55	91.74
[1,2,3,4]	94.73	81.57	27.91	88.87	93.79

(Count)

n	SVM	LR	kNN	NB	MLP
1	90.33	57.32	**86.89**	84.49	92.11
2	90.35	**59.01**	69.89	89.71	93.42
3	84.59	57.17	36.22	87.68	90.43
4	75.26	54.97	24.91	89.76	82.24
[1,2]	**92.4**	57.25	85.45	**89.83**	**93.91**
[3,4]	83.35	55.2	29.64	89.02	88.12
[1,2,3,4]	91.69	58.04	82.61	89.71	93.32

For durations, the music21 package was used to extract tuples consisting of musical event type (note, chord or rest) and the duration of the event. Each unique tuple is then encoded into a specific id, before tf-idf values are computed for all pieces. Only unigrams are considered, since the durations are note collected in exact sequential order and duration n-grams would not be an adequate representation of a piece. The computed tf-idf matrix is then merged with the previously computed chord n-grams tf-idf matrix. Table VII shows common duration types in the dataset; the complete feature set can be accessed in the source code[2].

III. THE CLASSIFIERS

A. Experiment Settings

As mentioned above, a total of 4,042 musical scores from 5 different composers were used. The dataset is skewed, with

[2]https://www.github.com/achimkoh/midi-classification

TABLE VII. TOP 10 DURATION TYPES IN DATASET

Duration Type	%
Eighth	25.24
Quarter	22.27
16th	18.34
Dotted Half	8.8
Half	5.72
Whole	4.38
32nd	3.42
Dotted Quarter	2.35
Eighth Triplet (1/3 QL)	2.2
16th Triplet (1/6 QL)	1.7

Bach accounting for more than 55% of the scores as opposed to Debussy accounting for less than 5%. A shuffled 10-fold cross-validation was used to split the full dataset into training and test data, and the sum of the ten confusion matrices is presented as well as the mean and standard deviation of F-measure metrics. The F-measure can be interpreted as an average of precision (π) and recall (ρ), with

$$\pi_i = TP_i/(TP_i + FP_i), \rho_i = TP_i/(TP_i + FN_i) \quad (1)$$

where TP_i is the true positive for class i, FP_i the false positive, and FN_i the false negative. F-measure values that are larger and closer to 1 indicate better classification quality. [17]

In a multiclass situation there exist several types of average that can be used to compute the overall F-measure. Here, the micro-average and macro-average are both presented to be more informative. The micro-average method computes metrics globally by counting the total true positives, false negatives and false positives. π and ρ are computed as follows:

$$\pi = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{M} TP_i}{\sum_{i=1}^{M}(TP_i + FP_i)},$$

$$\rho = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{M} TP_i}{\sum_{i=1}^{M}(TP_i + FN_i)}$$

where M is the number of classes. Then $F(\text{micro}) = 2\pi\rho/(\pi + \rho)$. It considers each individual result with equal weight, and therefore is highly influenced by the dominant classes.

The macro-average method computes metrics for each class and find their unweighted mean, which assumes equal priors and therefore is more sensitive to rare classes. For each class, π and ρ are computed as in Equation 1. Then F-measure for class i and the macro-averaged F-measure are computed as follows:

$$F_i = \frac{2\pi_i\rho_i}{\pi_i + \rho_i}, F(\text{macro}) = \frac{\sum_{i=1}^{M} F_i}{M}$$

where M is the number of classes. [17]

Scikit-learn is a Python library that provides a wide range of machine learning algorithms for both supervised and unsupervised problems [16]. Five different classifiers provided by scikit-learn were selected in this experiment. The modules LinearSVC, LogisticRegression, KNeighborsClassifier, MultinomialNB, and MLPClassifier are respectively implementations of a support vector machine with a linear kernel, a logistic regression model, a k-neighbors classifier, a Naive Bayes classifier for multinomial models, and a multilayer perceptron. Tree-based models were not considered since the data is sparse; while the feature dimension is very high, as shown in V, 75% of the pieces are shorter than 500 quarter notes in length, and 90% are shorter than 1000. Both unigrams and bigrams of chords were used as feature; only unigrams were used for durations.

TABLE VIII. CLASSIFICATION PERFORMANCE (USING CHORD N-GRAMS ONLY)

	SVM	LR	kNN	NB	MLP
F(micro)	0.9735	0.9463	**0.9179**	0.8884	0.9498
SD	(0.0073)	(0.0132)	(0.0126)	(0.0017)	(0.0285)
F(macro)	0.9652	0.9201	**0.8611**	0.8196	0.9226
SD	(0.0126)	(0.0215)	(0.0269)	(0.0363)	(0.0518)
Time (sec.)	8.21	0.95	2.49	0.13	21.60

TABLE IX. CLASSIFICATION PERFORMANCE (USING CHORD N-GRAMS AND EVENT DURATIONS)

	SVM	LR	kNN	NB	MLP
F(micro)	**0.9859**	**0.9463**	0.9077	**0.9218**	**0.9743**
SD	(0.0055)	(0.0206)	(0.0206)	(0.0174)	(0.0071)
F(macro)	**0.9841**	**0.9279**	0.8574	**0.8783**	**0.963**
SD	(0.006)	(0.0301)	(0.0325)	(0.0324)	(0.0091)
Time (sec.)	18.17	0.96	4.31	0.44	48.59

Table VIII shows the performance of the compared classifiers. SVM gives the best results, both in overall (as shown by its micro-averaged F-measure) and across classes (as shown by the macro-averaged F-measure). NB is the worst performer, and also displays the biggest discrepancy across classes as the difference between the two F-measure shows. Table IX shows the performance when also using duration features; using both feature sets improved performance and/or reduced the difference between the two metrics in all cases except kNN, which produced slightly worse results. The better results are marked in bold.

B. Support Vector Machine

A support vector machine (SVM) maps input vectors into a high-dimensional space and produces a separating hyperplane [18]. SVMs are considered to be effective with sparse data sets, and are commonly used in such fields as text classification along with features similar to this experiment [19] [20]. The LinearSVC module tested here is an implementation that uses a linear kernel and the LIBLINEAR[3] library as solver, and is set up as an L2-regularized L2-loss support vector classifier which solves the dual problem. Given a training set of $x_i \in R^n, i = 1, ..., l$ and binary class labels $y \in R^l$ such that $y_i = \{1, -1\}$, this classifier generates a weight vector w with the decision function $\text{sgn}(w^T x)$. It solves the following problem:

$$\min_{\alpha} \frac{1}{2}\alpha^T \overline{Q}\alpha - e^T\alpha$$

$$\text{subject to } 0 \leq \alpha_i, i = 1, ..., l.$$

[3]http://www.csie.ntu.edu.tw/~cjlin/liblinear/

TABLE X. MULTICLASS CONFUSION MATRIX
(SVM, USING CHORD N-GRAMS ONLY)

		Pred.					
		BA	BE	DE	SC	VI	ρ_i
True	BA	2231	5	0	5	4	0.9938
	BE	23	618	4	22	0	0.9265
	DE	3	2	192	1	1	0.9648
	SC	9	20	1	567	1	0.9482
	VI	0	1	0	5	327	0.982
	π_i	0.9846	0.9567	0.9746	0.945	0.982	

TABLE XI. MULTICLASS CONFUSION MATRIX
(SVM, USING CHORD N-GRAMS AND EVENT DURATIONS)

		Pred.					
		BA	BE	DE	SC	VI	ρ_i
True	BA	2236	5	1	2	1	0.996
	BE	25	639	1	2	0	0.958
	DE	1	3	195	0	0	0.9799
	SC	5	11	0	582	0	0.9732
	VI	0	0	0	0	333	1
	π_i	0.9863	0.9711	0.9898	0.9932	0.997	

where e is the all-ones vector, $\overline{Q} = Q + D$, D is a diagonal matrix, $Q_{ij} = y_i y_j x_i^T x_j$, $U = \infty$, and $D_{ii} = 1/(2C), \forall_i$ [21]. $C > 0$ is the penalty parameter and is set to 5. The currently presented implementation relies on a one-vs-all approach when dealing with multiclass classification.

Classification results for the SVM is shown in tables X and XI. It shows the best overall classification results, with a 5-class $F(\text{micro})$ score of .9859, as well as a stable performance across classes with an $F(\text{macro})$ score of .9841 when using both feature sets. Most misclassifications come from the BA v. BE and BE v. SC pairs. This tendency can be observed across classifiers, which points to potential similarities in the represented dataset.

C. Logistic Regression

Logistic regression a probabilistic linear classification model, where the class probability distribution is modeled using a logistic function [22]. The LogisticRegression module solves the following optimization problem [16]:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1)$$

and it is here set to use a stochastic average gradient descent solver, which can deal with multiclass classification.

Classification results for LR is shown in tables XII and XIII. The LR classifier is the third best performer among the compared classifiers, with a .9463 $F(\text{micro})$ score and .9279 $F(\text{macro})$ score. The bigger drop in $F(\text{macro})$ score than that SVM indicates a larger number of misclassification in specific classes, especially BE v. DE and BE v. SC. However, using both feature sets instead of only chord n-grams slightly improved this discrepancy: the $F(\text{macro})$ score went up from .9201 to .9279.

D. K-Nearest Neighbors

A nearest neighbors classification method relies on finding training data samples that are closest to the observed sample, and computing a majority vote from the respective classes of these nearest neighbors [16]. The KNeighborsClassifier module tested here uses k nearest neighbors for each test sample; k was set to 5 experimentally. The vote of each neighbor was weighted inverse-proportionally to the distance.

TABLE XII. MULTICLASS CONFUSION MATRIX
(LR, USING CHORD N-GRAMS ONLY)

		Pred.					
		BA	BE	DE	SC	VI	ρ_i
True	BA	2211	15	5	10	4	0.9849
	BE	39	585	7	31	5	0.8771
	DE	7	21	167	3	1	0.8392
	SC	13	37	5	541	2	0.9047
	VI	1	2	1	8	321	0.964
	π_i	0.9736	0.8864	0.9027	0.9123	0.964	

TABLE XIII. MULTICLASS CONFUSION MATRIX
(LR, USING CHORD N-GRAMS AND EVENT DURATIONS)

		Pred.					
		BA	BE	DE	SC	VI	ρ_i
True	BA	2176	33	3	25	8	0.9693
	BE	38	592	17	15	5	0.8876
	DE	2	18	176	3	0	0.8844
	SC	18	23	1	556	0	0.9298
	VI	3	5	0	0	325	0.976
	π_i	0.9727	0.8823	0.8934	0.9282	0.9615	

Classification results for kNN is shown in tables XIV and XV; it shows a five-class $F(\text{micro})$ score of .9179 and $F(\text{macro})$ score of .8611 when using only chord n-grams. It tends to have more difficulties classifying Debussy pieces than the previous classifiers; Beethoven pieces are also being misclassified as Bach, Scarlatti or Victoria. Such tendency is also evident in pairwise classification test results, which show lower $F(\text{macro})$ scores in the mentioned pairs. The kNN model did not benefit from using both feature sets; doing so improved performance on SC, but worsened on classes BE and DE.

TABLE XIV. MULTICLASS CONFUSION MATRIX
(KNN, USING CHORD N-GRAMS ONLY)

		Pred.					
		BA	BE	DE	SC	VI	ρ_i
True	BA	2225	6	1	6	7	0.9911
	BE	73	527	2	28	37	0.7901
	DE	21	42	120	11	5	0.603
	SC	41	13	0	508	36	0.8495
	VI	0	1	0	2	330	0.991
	π_i	0.9428	0.8947	0.9756	0.9153	0.7952	

TABLE XV. MULTICLASS CONFUSION MATRIX
(KNN, USING CHORD N-GRAMS AND EVENT DURATIONS)

		Pred.					
		BA	BE	DE	SC	VI	ρ_i
True	BA	2178	17	0	36	14	0.9702
	BE	138	507	1	9	12	0.7601
	DE	34	66	98	1	0	0.4925
	SC	38	6	0	554	0	0.9264
	VI	1	0	0	0	332	0.997
	π_i	0.9117	0.8507	0.9899	0.9233	0.9274	

TABLE XVI. MULTICLASS CONFUSION MATRIX (NB, USING CHORD N-GRAMS ONLY)

		Pred. BA	BE	DE	SC	VI	ρ_i
	BA	2178	51	3	5	8	0.9702
	BE	19	627	5	5	11	0.94
True	DE	2	84	111	1	1	0.5578
	SC	21	218	2	355	2	0.5936
	VI	2	11	0	0	320	0.961
	π_i	0.9802	0.6327	0.9174	0.9699	0.9357	

TABLE XVII. MULTICLASS CONFUSION MATRIX (NB, USING CHORD N-GRAMS AND EVENT DURATIONS)

		Pred. BA	BE	DE	SC	VI	ρ_i
	BA	2168	42	4	13	18	0.9657
	BE	22	628	7	4	6	0.9415
True	DE	1	65	133	0	0	0.6683
	SC	17	104	2	475	0	0.7943
	VI	3	8	0	0	322	0.967
	π_i	0.9806	0.7414	0.911	0.9654	0.9306	

TABLE XVIII. MULTICLASS CONFUSION MATRIX (MLP, USING CHORD N-GRAMS ONLY)

		Pred. BA	BE	DE	SC	VI	ρ_i
	BA	2206	17	1	18	3	0.9826
	BE	25	596	14	29	3	0.8936
True	DE	3	30	162	4	0	0.8141
	SC	11	24	9	552	2	0.9231
	VI	2	8	0	0	323	0.97
	π_i	0.9818	0.883	0.871	0.9154	0.9758	

TABLE XIX. MULTICLASS CONFUSION MATRIX (MLP, USING CHORD N-GRAMS AND EVENT DURATIONS)

		Pred. BA	BE	DE	SC	VI	ρ_i
	BA	2218	8	1	14	4	0.988
	BE	20	631	6	10	0	0.946
True	DE	2	11	183	3	0	0.9196
	SC	4	17	2	575	0	0.9615
	VI	1	1	0	0	331	0.994
	π_i	0.988	0.9446	0.9531	0.9551	0.9881	

E. Naive Bayes

The MultinomialNB module used here is a naive Bayes classifier that assumes a multinomial distribution. The Bayes theorem quantifies the conditional probability of a class variable given the knowledge about feature variables, with the assumption of conditional independence [22]:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

A naive Bayes classifier is based on this rule, and can be expressed as:

$$\hat{y} = \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

When using only chord n-grams, it is the worst performer among the 5 compared classifiers, and also the most inconsistent across classes, with .8884 $F(\text{micro})$ score and .8196 $F(\text{macro})$ score. (Table XVI) It is notable that a large number of Scarlatti and Debussy pieces are misclassified as Beethoven, which is also visible in the pairwise classification results. While all classifiers have more difficulty distinguishing between the two composers, the effect is especially drastic in NB; its BE v. DE and BE v. SC results are the only ones not to exceed .80 $F(\text{macro})$ score in in this experiment. However, by including durations as feature, NB shows the biggest improvement in terms of both metrics, and outperforms kNN; its performance on DE and SC are especially improved. (Table XVII)

F. Multilayer Perceptron

A perceptron is a classifier that computes a linear combination of inputs and returns a class [23]. A multilayer perceptron includes multiple layers of nodes in a directed graph, where each layer is fully connected to the next layer [24] [25]. The MLPClassifier module used here was set to use 10 hidden layers.

Classification results for the MLP is shown in Table XVIII. With .9743 $F(\text{micro})$ score and .963 $F(\text{macro})$ score, it is the second-best among the tested classifiers. It shows a similar confusion matrix to that of LR, with slightly better results. However, the large amount of time needed to train the classifier is a weakness. The bottom row of Table IX shows an overview of time required to fit training data and classify test data by each classifier.

IV. DISCUSSION AND FUTURE WORK

The experiments attempted to apply techniques common in text classification, such as tf-idf values and n-grams along with linear classifiers such as SVM, to music. The assumption is that chords are indicative of a composer's style and can be used similarly to words in a textual context. While the scope of the research is not wide enough to draw a conclusive statement, the good results obtained from some classifiers, especially the SVM, seem to indicate that this approach is worthy of further investigation. However, rather than relying on chord n-grams only, including the durations of musical events improved classification performance in most cases.

SVM is the best classifier, followed by MLP and LR. LR has the advantage of being fast, but it has more difficulties distinguishing specific composer combinations than SVM. This discrepancy is most acutely displayed in NB, of which other classification results are comparable to better performers but shows very poor performance when trying to guess between Beethoven and Debussy, or Beethoven and Scarlatti. This tendency was somewhat remedied by using durations as features, but this in turn worsened the performance of kNN. Future research could give a closer look on what characteristics of scores by these composers makes the classifiers behave this way, which would be helpful in building a better classification model.

Experiment results show that chord unigrams and bigrams featurized using tf-idf values are useful representation for music classification. N-grams with $n = 3$ or higher were not very useful, contrary to the first intuition, except for the Naive Bayes model. Whether this is due to the number of features becoming too high, or whether the parameters of feature extraction such as the length of time segment were

not adequate in capturing a piece's characteristics, could be addressed in future work. In addition, the feature extraction process was quite time-consuming and will benefit from a more efficient re-design. The current feature extraction method is also set up in a way that almost all other musical information is lost through the approximation to the closest chord, which might be a potential problem when trying to apply this method to more music-theoretical settings. Given that including duration improved classification performance, future work could explore combining the two feature sets instead of employing them separately.

By tracing back and examining more closely the often-misclassified pieces, one could gain more insight both in musical and computational terms. A casual look into the original MIDI files on misclassified pieces composed by Beethoven and Scarlatti, which showed bad performance when using chord n-grams only for all classifiers except kNN (Table XIV), revealed a formal difference in note durations that could not be captured by this experiment's initial feature extraction method, which ignored the duration of notes. An additional feature set considering the duration of musical events was devised in order to engage with this problem. By using both feature sets, all classifiers except kNN attained an improvement in performance. Future work could also consider other features than chords and duration, such as melody and base notes, as well as other methods of featurizing chords.

REFERENCES

[1] J. E. Youngblood, "Style as information," *Journal of Music Theory*, vol. 2, no. 1, pp. 24–35, 1958.

[2] D. Herremans, K. Sörensen, and D. Martens, "Classification and generation of composer-specific music using global feature models and variable neighborhood search," *Computer Music Journal*, vol. 39, no. 3, pp. 71–91, 2015.

[3] H.-T. Cheng, Y.-H. Yang, Y.-C. Lin, I.-B. Liao, and H. H. Chen, "Automatic chord recognition for music classification and retrieval," in *Multimedia and Expo, 2008 IEEE International Conference on*. IEEE, 2008, pp. 1505–1508.

[4] L. B. Meyer, *Style and music: Theory, history, and ideology*. University of Chicago Press, 1989.

[5] G. Buzzanca, "A supervised learning approach to musical style recognition," in *Music and Artificial Intelligence. Additional Proceedings of the Second International Conference, ICMAI*, vol. 2002, 2002, p. 167.

[6] A. D. Patel, "Language, music, syntax and the brain," *Nature neuroscience*, vol. 6, no. 7, pp. 674–681, 2003.

[7] A. Aizawa, "An information-theoretic perspective of tf–idf measures," *Information Processing & Management*, vol. 39, no. 1, pp. 45–65, 2003.

[8] J. Wołkowicz, Z. Kulka, and V. KEŠELJ, "N-gram-based approach to composer recognition," *Archives of Acoustics*, vol. 33, no. 1, pp. 43–55, 2008.

[9] R. Hillewaere, B. Manderick, and D. Conklin, "String quartet classification with monophonic models." in *ISMIR*, 2010, pp. 537–542.

[10] C. Hauser, "Probabilist modeling of musical chord sequences for music analysis," 2009.

[11] M. Ogihara and T. Li, "N-gram chord profiles for composer style representation." in *ISMIR*, 2008, pp. 671–676.

[12] B. Absolu, T. Li, and M. Ogihara, "Analysis of chord progression data," in *Advances in Music Information Retrieval*. Springer, 2010, pp. 165–184.

[13] M. Li and R. Sleep, "Melody classification using a similarity metric based on kolmogorov complexity," *Sound and Music Computing*, vol. 2012, 2004.

[14] B. Manaris, J. Romero, P. Machado, D. Krehbiel, T. Hirzel, W. Pharr, and R. B. Davis, "Zipf's law, music classification, and aesthetics," *Computer Music Journal*, vol. 29, no. 1, pp. 55–69, 2005.

[15] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," 2010.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[17] A. Özgür, L. Özgür, and T. Güngör, "Text categorization with class-based and corpus-based keyword selection," in *International Symposium on Computer and Information Sciences*. Springer, 2005, pp. 606–615.

[18] V. Vapnik, *The Nature of Statistical Learning Theory*, ser. Information Science and Statistics. Springer New York, 2013. [Online]. Available: https://books.google.com/books?id=EqgACAAAQBAJ

[19] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision."

[20] T. Joachims, *Text categorization with Support Vector Machines: Learning with many relevant features*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 137–142. [Online]. Available: http://dx.doi.org/10.1007/BFb0026683

[21] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[22] C. C. Aggarwal, *Data mining: the textbook*. Springer, 2015.

[23] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.

[24] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," DTIC Document, Tech. Rep., 1961.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," DTIC Document, Tech. Rep., 1985.