Music Composition and Classification with n-gram Based Markov and Naive Bayes Models

Hannah Phan, Jessica Tian, Nina Uzoigwe, Lan Zhang

Abstract

This project aims to compose musical pieces using Markov-Chain models trained on a piece of the same genre, and to classify the genre of a given piece using a n-gram based Naive Bayes model. Our results suggest that generating pieces using higher-order Markov models results in compositions that sound stylistically similar to the training piece and reflect their musical motifs. Additionally, using a Naïve Bayes model trained on bi-grams derived from a corpus of Baroque and Romantic pieces, we were able to achieve an overall test accuracy of 93.33% on identifying Baroque pieces and 92.31% on identifying Romantic pieces.

1 Introduction

Constructing algorithms to compose music with minimal human interventions has had a long history. In fact, in the 18th century, a music game called Müsikalisches Wurfelspiel was developed, where small fragments are randomly combined by chance, often done by tossing dice[1]. A wide variety of probabilistic models and learning algorithms have been utilized for automated composition, including Markov models[2]. More recently, Markov chains have been built from previous music pieces of different genres and can be the basis of different learning algorithms to make probabilistic decisions and create new music pieces in the same genre[3,4].

Conceptually, Markov chains encode the probability of moving from one given state to another in a sequence of events, with the probability of the next state dependent only on the current state. They can be applied easily to musical compositions since a piece can be interpreted as a sequences of states, where a possible definition for a state is a note or chord with a specified time duration. Different orders n of Markov chains over some state space S is equivalent to a first order Markov chain over all n-length tuples of states in S. In other words, a first order Markov Chain is a stochastic model that transitions from one state to the next, where the probability of each future state depends only on the current state. Markov chains of second or higher order are processes in which transition probability of the next state depends on two or more preceding ones. The system contains a finite set of states, has a designated initial state, and moves in discrete steps according to transition probabilities.

The goal of this report is two-fold. First, we develop first-order Markov chain models each trained on a series of notes extracted from a violin solo, calculating the transition probabilities between notes through empirical approximation, by counting and normalizing the frequencies of each pairwise note transition in the training piece. Upon supplying an initial sequence of notes, we use the trained Markov chain to generate a new piece stylistically similar to the parent piece. We then extend the model to include second-order to fourth-order Markov chains and compare the quality of the generated compositions across orders through

[1] Cope, D. (1996). Experiments in Musical Intelligence. A-R Editions, Inc.
[2] https://www.worldscientific.com/doi/pdf/10.1142/S2010194512007829
[3] https://dke.maastrichtuniversity.nl/gm.schoenmakers/wp-content/uploads/2015/09/Linskens-Final-Draft.pdf
[4] https://www.jstor.org/stable/1575226?seq=1

both quantitative and qualitative metrics. The second part of this report is to create a musical genre classification model using a Naive Bayes classifier trained on n-grams of a corpus of pieces. Successfully constructing musical pieces through computational means has significant implications for digital music, as many automated composing systems are currently being developed[5], and being able to classify the genre of a given piece with a high accuracy is useful for music recommendation and streaming softwares.

This paper is organized as follows. In Section 2, we will propose our Markov chain and Naive Bayes models. Dynamic analysis of the generated music and results from the genre classification model will then be examined in Section 3. Our findings from these models along with a sensitivity analysis of our parameters will be discussed in Section 4, with a conclusion following in Section 5.

2 Description of Models

2.1 Music Composition

In order to generate new compositions from a training piece, our model relies on a number of functions. The main function `compose` takes in two inputs, (1) a MIDI file corresponding to the preprocessed version (where the solo/melody part has been extracted) of a training piece along with (2) a value n that determines the order of the n-gram model (or equivalently the $(n-1)$-order Markov Chain) used to produce the composed piece[6]. We chose to use n-gram models to compose musical pieces because such models have been successfully used in language models for pattern recognition and text prediction purposes, predicting the next item x_i based on the previous $x_{i-(n-1)}, ..., x_{n-1}$ items with the assumption that a given word's probability of appearing in a text depends only the last $n-1$ words. We can draw a correspondence between text and music, where words correspond to notes, and use n-gram models to approximate the true underlying rules of note evolution. Additionally, numerous studies and algorithmic composition softwares have successfully used Markov Chains of various orders to compose musical pieces, and a n-gram model is equivalently a $(n-1)$-order Markov Chain.

First, to process a MIDI file, or Musical Instrument Digital Interface file, which contains information regarding what and how notes are played[7], the function `compose` initially reads the input MIDI file and records the notes, their velocities, their time durations and their respective musical channels (corresponding instruments) in a list. In this first step, the tempo of the original training piece along with its lowest note and smallest note duration length are recorded in variables called `BPM`, `LOWEST_NOTE`, and `BASE_TICK` respectively. The helper function `midi_to_unigram` then appends each note in the input MIDI file to a list called `current_states` m number of times, where m encodes the duration of the note. We determine m for each note based on how many times the `BASE_TICK` divides into the given note's original duration (or more formally, m = duration/`BASE_TICK`), so that if a note is playing for 20 ticks and the base tick of the piece is 4, we would append the same note 5 times to `current_states`. Doing so removes the need for creating another data structure that stores the time durations of each note in the training piece. We then take the length of `current_states` to record the number of beats in the piece (and number of notes to generate in the composition) as well as the set of notes in the piece, under the variable `vocabulary`.

In the next step, we get a count of all the n-grams in the input piece, by first using the `ngrams` function on the `vocabulary` of notes to return all possible n-length tuples created from elements in the `vocabulary` set, before then using the function `ngram_counts` to return a dictionary containing the frequency of each

[5] https://alternativeto.net/software/automated-composing-system/

[6] Midi Files were primarily source from here and here

[7] Tim Fischer, 2020, "What Is a MIDI File?", https://www.lifewire.com/midi-file-2621979

n-gram that occurs in the inputted piece. Next, we build a n-gram model (using the function `ngram_model`) from the n-gram counts returned in the previous step, in the form of a dictionary containing the transition probabilities from one n-gram to another n-gram in the training piece. This is where the transition matrix is created. In order to more easily compose pieces in the next step, we store all n-grams, in both the dictionary returned by `ngram_counts` and that returned by `ngram_model`, with a context and a target, where the context is the first $n-1$ notes and the target is the last nth note in the n-gram.

We then compose a new piece of music using the transition probabilities from the training piece. The function `sample`, given a n-gram model that contains all the transition probabilities and a context (initial sequence) of notes of length $n-1$, produces a next note by randomly generating a probability and finding the next note whose transition probability from the context corresponds with this generated probability. For example, if we are using a trigram model $n = 3$, the supplied context will need to be 2 notes and the `sample` function will return the third note. `sample_sequence` samples from the model `count` number of times using the `sample` function and a `start_context`, which is the first $n-1$ non-silent notes of the training piece, before updating the context at each timestep to include the newly generated note. When `sample_sequence` is run, it returns a list of unigrams corresponding to the generated piece. Then the list of unigrams, or the composition, is passed to `unigram_to_midi`[8] which outputs a MIDI file, saving the file to the hard drive. As previously mentioned, our main function `compose` ties together all the aforementioned helper functions by taking in a pre-processed MIDI file and a `n` value for the length of the n-grams, before training a `n - 1` order Markov model based on the note transitions of the training piece and returning a new composition with the file name of the generated piece.

2.2 Music Classification

We used a bi-gram Naive Bayes classifier with Laplace smoothing as the principal classifier for this work. This method of classification was applied to (1) genre classification (Baroque vs Romantic) and (2) within-genre composer classification (Chopin vs Paganini). This approach was taken because we expect note transitions and note vocabulary to be different across genres and composers. In general, this probabilistic classifier is typically used when performing text classification tasks such as determining authorship. Due to the similarity between words in text and notes in music, we selected this approach to classification. In the particular case of music classification, we looked at the conditional probabilities, or likelihoods, of bi-gram note transitions in a piece. It is important to note that using a model like Naive Bayes requires the independence assumption to be true among individual probabilities. While this is generally not the case in music nor text, the classification performs surprisingly well, as we see from our results as well as from preexisting literature.[9,10]

$$L(w) = \frac{n_w(D_t) + \alpha}{n_o(D_t) + \alpha|V|} \tag{1}$$

An individual bi-gram's smoothed likelihood is defined above in Equation 1. On the left hand side, we have the likelihood of bi-gram w appearing given some training corpus D_t. $n_w(D_t)$ is the number of occurrences of bi-gram w in the training corpus. $n_o(D_t)$ is the total number of bi-grams in the training corpus.

With the addition of the smoothing parameter α, we are able to prevent the likelihoods of a sequence of bigrams from going to 0. $|V|$ is the cardinality of the lexicon (i.e., the number of distinct bi-grams in

[8] Add credit for this function

[9] https://www.computer.org/csdl/pds/api/csdl/proceedings/download-article/12OmNzcPA6v/pdf

[10] http://cs229.stanford.edu/proj2018/report/18.pdf

the training corpus). Laplace smoothing is a technique that addresses the problem of zero probability in Naive Bayes, which occurs when a bi-gram in a test piece is not included in the training corpus. In such a scenario, without smoothing, $L(w) = \frac{n_w(D_t)}{n_o(D_t)}$ would equal 0. Pointing to Laplace's rule of succession, some authors argued that α should be set to 1, which is also known as add-one smoothing.[11,12] Thus, for our classifications, we selected $\alpha = 1$.[13]

Putting together individual bi-gram likelihoods, the likelihood of a new sequence H is simply defined by the likelihood of each token, multiplied by each other:

$$L(H) = \prod_{w \in H} L(w) \tag{2}$$

Since multiplying a string of small probabilities resulted in an extremely small likelihood value, we opted to take the log likelihood instead.

$$l(H) = \sum_{w \in H} \ln\left(L(w)\right) \tag{3}$$

2.2.1 Genre Classification

Applying the Naive Bayes methodology to genre classification, we calculated (1) the likelihood of the bi-gram sequence H of a test piece given a train corpus of Baroque pieces, $l_B(H)$, and (2) the likelihood of the bi-gram sequence H of a test piece given a train corpus of Romantic pieces, $l_R(H)$. If $l_B(H) > l_R(H)$ for a piece in the test set, we then classified it as Baroque. Otherwise, we classified it as Romantic.

We primarily focused on gathering pieces in A minor and its relative major C major, although there are a couple of pieces that are in alternative key signatures. We then performed a randomized 60/40 train-test split on 36 Baroque pieces and 31 Romantic pieces, resulting in a train set of 21 Baroque pieces and 18 Romantic pieces.[14]

While there were more Baroque pieces, the Romantic pieces were, on average, longer and contained a larger number of note transitions, resulting in the Romantic training set having 49.9% more bi-grams. To address this concern, we experimented with two additional modifications to the smoothed Naive Bayes classifier: (1) adding weights such that the bi-grams in each piece of the training set are represented by the same amount in the final train corpus, and (2) splitting all of the pieces into segments of 2000 states (`current_states` = 2000) before performing the train-test split and Naive Bayes classification.[15]

Finally, the results from the Naive Bayes classifier (smoothed vs unsmoothed) were then compared to a baseline model that simply predicts the majority class from the training data set.

2.2.2 Composer Classification

Aside from across genre classification, we wanted to see if the Naive Bayes methodology could be applied to within-genre classification across different composers. For this case study, we opted to compare Chopin and Paganini, two Romantic-period composers with distinctive styles. Similarly, we performed a 60/40

[11] Jurafsky, Daniel; Martin, James H. (June 2008). Speech and Language Processing (2nd ed.). Prentice Hall. p. 132.
[12] Russell, Stuart; Norvig, Peter (2010). Artificial Intelligence: A Modern Approach (2nd ed.). Pearson Education. p. 863.
[13] Sensitivity analysis on the effect of α can be found in Section 4.2.1
[14] Refer to the Appendix to see which pieces were in the train and test sets.
[15] This resulted in a data set of 350 Baroque snippets and 643 Romantic snippets.

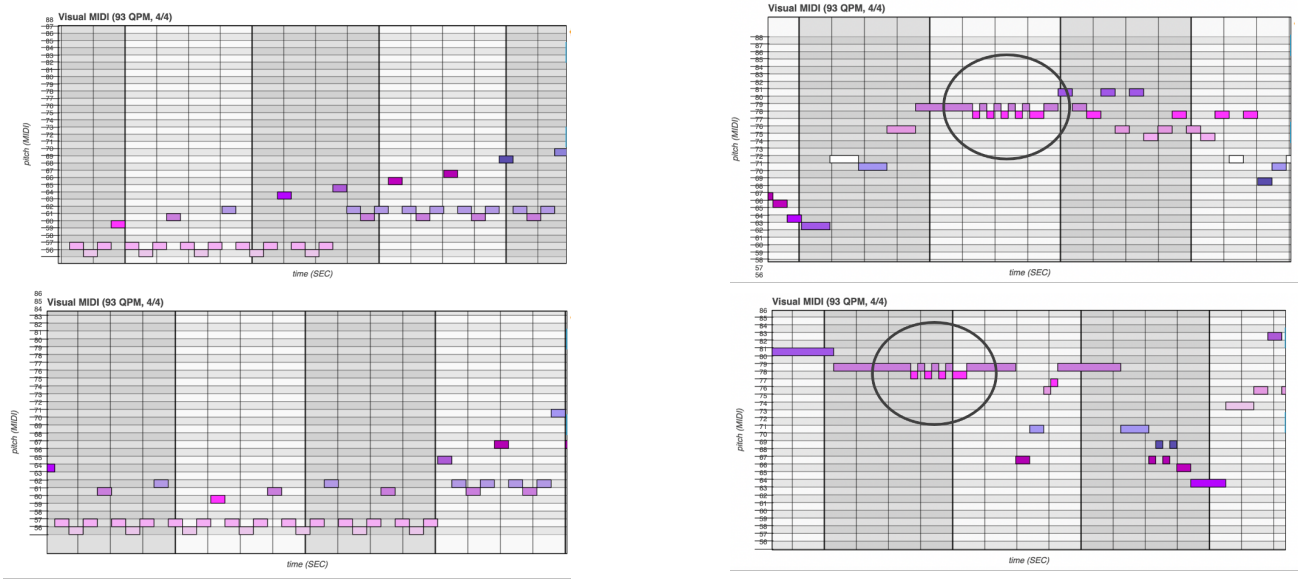
train-test split on 16 Chopin pieces and 9 Paganini pieces, resulting in a train set of 7 Chopin pieces and 5 Paganini pieces.[16] Like with genre classification, we compared the results to a baseline model that predicts the majority class from the training data set, in our case Chopin.

3 Results & Analysis

3.1 Music Composition

To qualitatively measure the generated compositions from our Markov models of varying order, we listened to the outputted pieces and visually examined the pitches in the associated MIDI file, focusing on Bach's Violin Concerto in A Minor as the training piece. For a first-order Markov chain model (2-gram model), where the predicted next note in a sequence depended solely on the previous note, we observe that the outputted pieces contain short note durations as well as disordered note sequences with no melodic phrasing even within small groups. However, some short trilled sequences (rapid alternating pitches/notes) from the original piece were mimicked in compositions by the 1st order Markov model. On the other hand, when we scale up to a fourth-order Markov model, we observed many motifs from the original training piece captured by the composition. For example in Figure 1 below, we observe recurring triplets breaking up a scaled passage (displayed on the left side) as well as ornamentation such as trills (displayed on the right side) and grace notes typical of Baroque music. However, across all orders of our Markov model, the composed pieces had little global structure of long melodic phrasing as displayed in the original piece, since a Markov chain only predicts the next note based on the values of a small collection of previous notes, giving it ability to only compose successfully on a local scale.

Figure 1: Motifs from the original piece (top) captured by the 4th-order Markov model composed pieces (bottom) for Bach's Violin Concerto in A Minor. The left hand graphs show the recurring triplets breaking up a scaled passage and the right-hand visuals show trilled passages.



To quantitatively measure generated compositions from the aforementioned Bach training piece, we pursued musicality, originality, and human measures. We chose these metrics based on the evaluation criteria of a similar previous study used to compose music using Hidden Markov Models (HMM)[17]. To measure

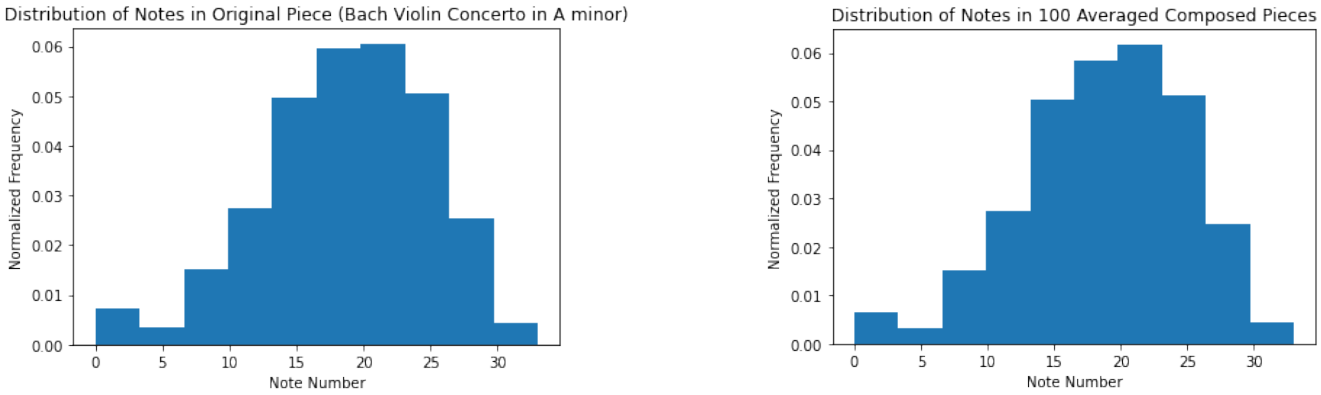
[16] Refer to the Appendix to see which pieces were in the train and test sets.
[17] https://arxiv.org/pdf/1708.03822.pdf

musicality, first we chose to calculate the distributions of consonant (perfect) and dissonant (imperfect) intervals by finding the absolute differences between consecutive notes in the composed and original pieces before mapping these differences to a pre-defined interval[18]. Perfect intervals are essentially a pair of notes that when played together, sound perfect or resolved, while imperfect intervals sound unpleasant and tension-producing. As defined by music theorists, dissonant intervals are minor and major 2nds, tritones or augmented fourths, as well as minor and major 7ths, while some consonant intervals are major thirds, minor thirds, perfect fourths, and perfect fifths fifths. We calculated the proportions of dissonant intervals as well as the aforementioned consonant intervals in both the original and composed pieces by a fourth-order Markov model, averaging the proportions across 100 composed pieces to account for the randomness associated with composition.

Upon doing so, we found that the proportion of dissonant intervals in the composed pieces was approximately 0.25466, the proportion of thirds was 0.06355, and the proportion of fourth and fifths intervals was 0.05452, which matched the distribution of consonant versus dissonant intervals in the original piece, found to be 0.25446 for dissonant intervals, 0.06334 for thirds, and 0.05424 for fourths/fifths. It is important to note that in the original piece, a dissonant interval may be resolved by a fourth or fifth at later time steps, but our calculated metric of interval distributions gives no information on the temporal resolution of dissonant intervals in the original/composed pieces.

Second, we plotted the distribution of notes between the original and 100 averaged composed pieces, normalizing by length of the piece. In Figure 2 below produced for a fourth-order Markov chain, we observe that the distributions of notes match; we also observed, as expected, the same distribution of notes between the original versus the composed pieces across all orders of our Markov chain model.

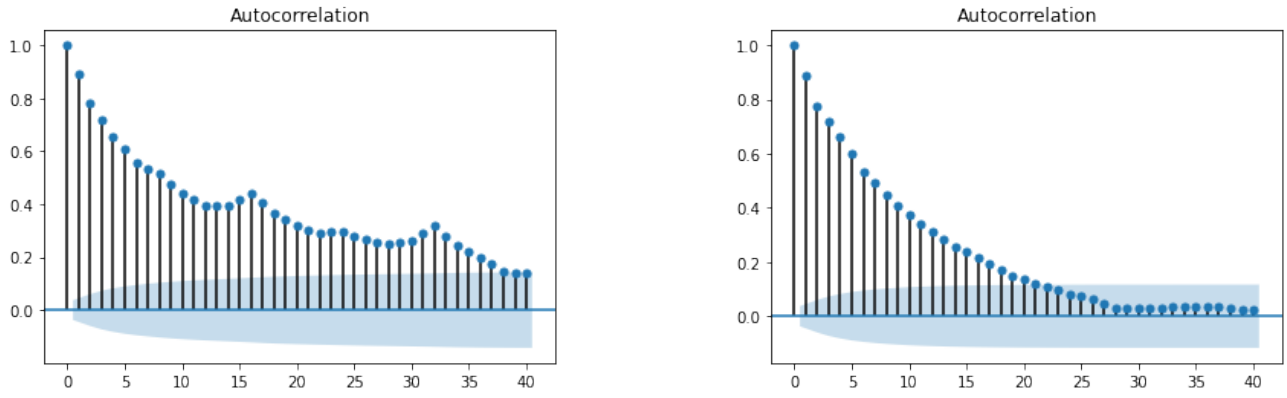
Figure 2: Distribution of Notes in Original Piece (Left) vs. 100 Averaged Composed Pieces (Right)



Third, we calculated the auto-correlation functions of the original and composed pieces to identify recurring melodies, which would give insight into a larger global structure of the piece. The auto-correlation function γ is often used in time series analysis, and is defined as the correlation of a signal with a copy of itself at a later time step, as a function of the time delay or lag h, formally defined as $\gamma(h) = \text{Corr}(x(t+h), x(t))$. In Figure 3, we calculate the auto-correlation function for the original and a generated piece out to a lag of $h = 40$; we observe at later time steps in the original piece, there is still some correlation with the initial parts which indicate a global musical structure. But, in the composed pieces for a fourth-order Markov chain, the correlation decays dramatically.

[18] https://en.wikipedia.org/wiki/Interval_(music)

Figure 3: Autocorrelation Plots between Original Piece (Left) vs. A Composed Piece (Right)



The shaded blue region is the confidence interval where if the bar exceeds the blue region, it is not a statistically significant correlation. This suggest that at larger time steps, there is less likelihood for a musical pattern in the beginning to reoccur. We also observe that the decay in correlation for the composed pieces generated by our first-order Markov model experiences an even faster decay towards 0 than the fourth-order model, as expected. Thus, it appears that our model retains local structures as we increase the order, but has difficulty capturing the global patterns that occur in the original piece.

To measure originality, we calculated the minimum edit distance across the 100 composed pieces by the first and fourth-order Markov models, which is the number of insertions, deletions and swaps to convert the sequence of notes associated with the original piece to the composed. We saw that for the first order model, the average edit distance across 100 compositions was 2357.81, while the fourth order model had a slightly lower average edit distance of 2303.62 across 100 compositions. To contextualize these edit distances, both the original and the composed pieces contained 2748 notes.

We noticed that these metrics were more confirming that our Markov models had correct transition probabilities and were overall constructed correctly than they were a telling measure of whether the generated pieces sounded "real" or "human". Moreover, none of the metrics effectively specify how the order of the chain affects the quality of the composed pieces, so we added a human evaluation with a blind survey and asked respondents to rank seven Baroque compositions (trained on Bach's Violin Concerto in A Minor) and six Romantic compositions (trained on Paganini's Caprice No. 24) on a scale from 1 to 5, where 1 is human composed and 5 is Markov Chain composed. When we averaged the 15 responses, we found that for the Baroque compositions, Sample 1, which was a fourth-order model composition, was rated as the most "Human" with a score of 1.8 and sample 4, which was a first-order Markov composition, was rated the most "Markov Chain" with a score of 4.66. Generally, we see a decreasing trend in the average ratings, indicating that listeners believe the composed pieces sound more "Human" as the order of the chain increases. Interestingly, we also observe this trend amongst the Romantic era compositions, however examining the distributions of the survey responses, we noticed that the distributions were less skewed towards "Human" or "Markov Chain" amongst the Romantic compositions. These results indicate that Baroque pieces, due to their more regular intervals and note patterns, are easier to compose from for a Markov model than Romantic pieces with greater variation in note range/chromaticism as well as time durations, and fewer recurrences of melodies throughout the piece.

Genre	Baroque							Romantic					
Model Order	1st		2nd	3rd	4th			1st		2nd	3rd		
Listening Order	4	6	3	2	7	1	5	1	5	4	2	3	6
Average Rating	4.667	3.867	4.000	2.467	1.8	3.133	2.533	4.2	4.133	3.867	2.8	3	2.867

Table 1: This table displays the average human ratings for all 13 Markov generated compositions, where 1 is Human and 5 is Markov Chain.

3.2 Music Classification

3.2.1 Genre Classification

For the baseline model, we predict Baroque since it is the majority class in our data set. We also compare the results of unsmoothed Naive Bayes with smoothed Naive Bayes. As mentioned in Section 2.2.1, we also experimented with 2 additional modifications to the smoothed Naive Bayes classifier:

- (Mod. 1) Weights were added such that the bi-grams in each piece of the training set were represented by the same amount in the final train corpus.
- (Mod. 2) All of the pieces were split into segments of 2000 states prior to the train-test split.

Model	Test Accuracy Scores		
	Baroque	Romantic	Overall
Baseline Model	100%	0%	53.57%
Naive Bayes (unsmoothed)	86.67%	7.69%	50.00%
Naive Bayes (smoothed)	93.33%	92.31%	92.86%
Naive Bayes (Mod. 1)	100%	53.85%	78.57%
Naive Bayes (Mod. 2)	74.29%	60.00%	67.14%

Table 2: This table displays the test accuracy scores for all 5 models. The Naive Bayes classifier with smoothing has the highest overall test accuracy score of 92.86% and performs the best out of all the models on Romantic pieces.

However, as we can see from Table 1, neither of the modifications improved the test accuracy score and instead decreased it. Furthermore, from Table 1, we see that without smoothing, the model severely over-predicts Baroque, much like the baseline model.

3.2.2 Composer Classification

Model	Test Accuracy Scores		
	Chopin	Paganini	Overall
Baseline Model	100%	0%	69.23%
Naive Bayes (unsmoothed)	0%	100%	30.77%
Naive Bayes (smoothed)	100%	100%	100%

Table 3: This table displaces the test accuracy scores for all 3 models.

As we can see from Table 2, the Naive Bayes classifier with smoothing classified all the pieces in the test set to be Chopin or Paganini correctly. This significantly outperforms both the baseline model and Naive

Bayes classifier without smoothing. Interestingly, the Naive Bayes model without smoothing over-predicts Paganini, perhaps because the Paganini pieces in the training set have significantly more unique bi-grams in comparison to the Chopin pieces in the training set.

4 Discussion

4.1 Music Composition

4.1.1 Sensitivity Analysis Using Different Order Markov Chains

We experimented with different orders of Markov Chains, starting with a first-order Markov Chain (a bi-gram model) as a baseline model for composition before incrementing to a fourth-order Markov Chain (a 5-gram model). Qualitatively, we observe that as we increase the order of the Markov Chain, the composed music has greater phrasal structure, picking up on some motifs, stylistic elements, and note durations of the training piece, whereas the baseline first-order Markov Chain produces pieces that sound like a collection of random notes with irregular durations. However, like all higher order Markov models, the empirical estimation of the model parameters (the transition probabilities) is significantly less reliable because we have fewer examples of transitions from one state to the next in the training data. Since each composed piece was produced by a Markov model whose transition probabilities were calculated based on the frequencies of these transitions in the original piece, our training data is sparse at higher orders, as there are fewer examples of transitions from a group of n notes where $n > 1$ to another group of n notes in the parent piece, versus transitions from a single note to another note used in a first-order model. Additionally, if the parent piece has fewer note transitions due to a simpler melody, such as a lullaby, the composed pieces from the resulting higher-order Markov Chains displayed significantly fewer note transitions than those from lower-order models.

This limitation became more evident when we experimented with 1-smoothing in our composition models, adding a frequency of 1 to all 5-gram transitions in a 4th order Markov Chain trained on Bach's Violin Concerto in A Minor. Doing so allowed us to account for the sparsity of transitions in the training dataset. However, upon using this 1-smoothed 4th order Markov Chain to generate original compositions, the results sounded even more disorderly than those of the 1st order chain, as if random notes were being played at each timestep. This result makes sense since we started with very low frequencies of a select number of 5-gram transitions, with most 5-gram transitions having a frequency (and thus likelihood) of 0 in the original training piece. Thus, if a certain 5-gram A only transitioned to another 5-gram B in the training piece and not to any other 5-grams, the row of the transition matrix corresponding to A would have a probability of 1 in the column associated with B and zero elsewhere. In a 1-smoothed model however, the transition probability from A to B would be $2/n$ where n is the total number of possible 5-grams in the training piece, while the transition probability from A to all other 5-grams would be $1/n$. This result is mirrored qualitatively in the generated pieces; with the unsmoothed 5-gram model, we see that the composed piece contains small note groups that also appear in the training piece due to the limited number of transitions from one group of notes to the next, however with the smoothed 5-gram model, all transitions were nearly uniform in probability, resulting in a jumbled generated piece. Additionally, a n-gram model grows exponentially in size with respect to n, and we faced memory issues when constructing fourth-order Markov Chain models (5-gram models) for longer training pieces with greater pitch variation.

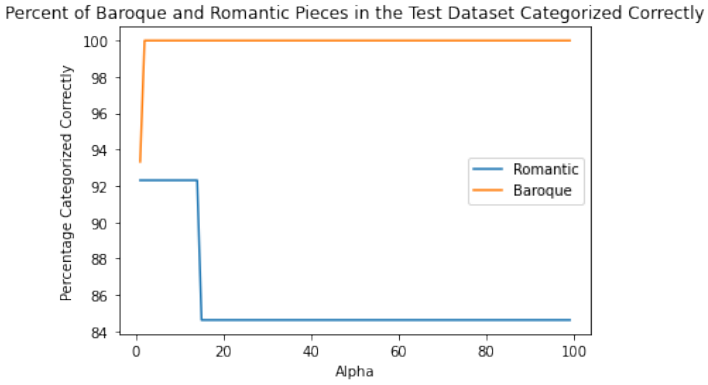
4.2 Music Classification

4.2.1 Genre Classification: Sensitivity Analysis on the Smoothing Parameter Value

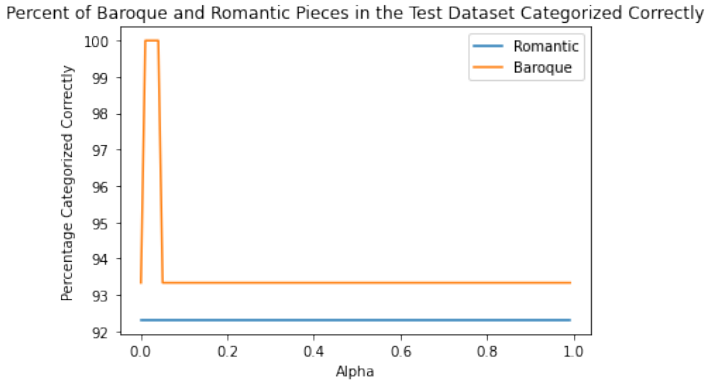
Let us take a look at how varying α, the smoothing parameter, in the Naive Bayes classifier changes our results for genre classification (Baroque vs Romantic).

In Figure 4, when $\alpha < 0.05$ and $\alpha \geq 2$, we observe that model classifies 100% of Baroque pieces correctly. When $\alpha \geq 15$, we see that the test accuracy for Romantic pieces decreases by 7.69%, suggesting that 1 additional Romantic piece was misclassified.

Figure 4: Effects of varying α on Baroque and Romantic Test Accuracy



(a) Zoom-out: α from 1 to 100



(b) Zoom-in: α from 0.01 to 1

In general, when selecting a value for the smoothing parameter, it makes sense to select an α that is less than or equal to 1. Furthermore, it seems that much of the existing literature follows this rule of thumb. A large α is not ideal because the larger the α, the less important the information provided by the training corpus becomes. Since we have many states, $n_w(D_t)$ is generally very small. As the result, for large α, $L(w) \approx \frac{\alpha}{n_o(D_t)+\alpha|V|}$. This suggests that the ideal α value lies in between 0.01 and 0.05; however, it is unclear if this would still hold if the training set is expanded. Overall, surprisingly, varying α from 0.01 to 100 did not have a large effect on the test accuracy score as the number of correctly classified Baroque and Romantic pieces only fluctuated by ± 1. This demonstrates the robustness of the smoothed Naive Bayes approach, in comparison to unsmoothed Naive Bayes, which performed significantly worse.

4.2.2 Genre Classification: Varying The Training Set

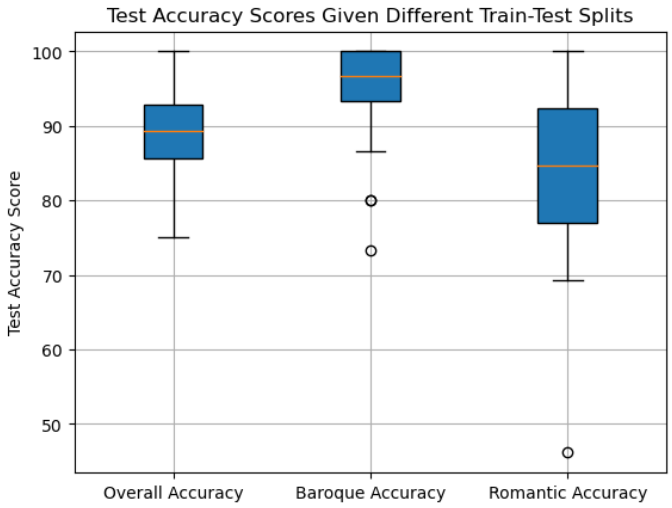
Let us see how changing the random seed for the 60/40 train-test split in our genre classification changes our test accuracy score results given the smoothed Naive Bayes model from Section 3.2.1.

In Figure 5 below, we see that with the exception of one outlier, the Naive Bayes classifier with smoothing outperforms all four of the other models in Section 3.2.1.[19] Additionally, in Figure 5, it seems that the model's performance on Romantic pieces is highly variable, with a variance that is over 3x that of Baroque pieces. This makes sense because note transitions in Romantic pieces are generally less structured than those that appear in Baroque pieces. As the result, the performance of our model on the test set is largely dependent upon how representative the pieces in the training set are of those in the test set.

[19] The models are the 1) Baseline Model, 2) Unsmoothed Naive Bayes, 3) Smoothed Naive Bayes (Mod. 1), and 4) Smoothed Naive Bayes (Mod. 2)

Overall, while the smoothed Naive Bayes classifier generally performs exceptionally well, it is important to note that its overall performance depends largely on the train-test split.

Figure 5: Test Accuracy Scores for 20 Different Train-Test Splits



Model	Test Accuracy Scores		
	Baroque	Romantic	Overall
Mean	94.00%	82.69%	88.75%
Standard Deviation	7.86%	11.63%	6.51%

4.2.3 Composer Classification: Adding a Baroque Composer

In addition to classifying pieces to be Chopin or Paganini, we tested the impact of adding Baroque composer Bach to the mix, increasing the number of unique classes to 3. 24 pieces by Bach were included in the model and split 60/40 across the train and test sets. Surprisingly, when we add Baroque composer Bach, all the Chopin pieces in the test set are predicted to be Bach by the Naive Bayes classifier with smoothing. Additionally, all the Bach pieces were predicted as Paganini.[20] Thus, it appears that the smoothed Naive Bayes classifier performs significantly better when predicting across only two composers, in particular Chopin vs Paganini.

Overall, the main limitation of our music classification is the size of the dataset. With a small training set, idiosyncrasies within the pieces of the training set are overemphasized. By expanding the training set, we may be able to reduce the likelihoods on unusual, or rare, note transitions, which could further improve the Naive Bayes classifier's test accuracy score.

5 Conclusion

On the composition side, upon using varying orders of Markov chains trained on a given piece of music, we were able to successfully compose new pieces in the same genre, as evidenced by our human evaluation survey. We observed that with higher-order Markov chains, the composed pieces reflected a subset of the motifs and ornamentation present in the original training piece, though across all orders of our Markov

[20] Varying α did not change this outcome.

model, the composed pieces had little global structure. Additionally on the classification side, with a smoothed Naive Bayes model trained on 60% of the dataset, we were able to successfully classify the genre of the pieces (Baroque vs Classical) and the composer of the pieces (Paganini vs Chopin). While our results were exceptional when applied to 2 classes, as we saw with the addition of Bach to our composer classification, adding an additional class resulted in subpar results.

To expand our current models, several paths for future work can be taken. It would be of interest to expand the initial dataset and pieces that are used for training the Naive Bayes Classification Model. This can further include thousands of pieces and hundreds of composers from different musical genres outside of the Baroque and Romantic periods. It is essential to delve deeper into what makes a good training set, questioning whether it is important to have representation of all composers or only including specified pieces that fall under technical specifications. Secondly, for the Markov chains used in the composition portion of this project, we can explore different definitions for a state, perhaps using joint states that combine information about note pitch, tempo, and duration, as doing so may increase the quality of compositions and lead to pieces that are more humanistic in nature. For both the composition and genre classification models, it could be of interest to use more complex Neural Network models such as Long Short-Term Memory (LSTM) or Recurrent Neural Networks (RNN) that more accurately capture time series information, which would allow us to better identify the temporal qualities and global structure of training pieces during composition and genre classification.

6 Attribution of Effort

- Lan contributed to writing the code associated with composing new pieces using varying orders of Markov Chain Models (`Composition.ipynb`), generated quantitative metrics for these compositions as well as their visualizations (`Quantitative Metrics.ipynb` and `Midi File Visualizations.ipynb`), selected pieces for the human survey, and designed the survey. She also contributed to this writeup in the model description, results, and discussion sections, and to the creation of the presentation slides associated with the composition portions of this project.
- Nina contributed by conducting background research and finding publications on potential avenues for the project. When it came to analyzing the composed pieces, she also identified several potential metrics that could be implemented to quantitatively measure the accuracy of the composed pieces. She played an integral role in designing the human survey and distributing it to members of the Harvard community. She contributed by writing parts of the introduction as well as the conclusion and played a role in creating and presenting the slide deck associated with the paper, expanding on the background and motivation and the future works slide.
- Jessica contributed to the code associated with preprocessing (`preprocessing.ipynb` and `bach-prepocessing (with additional EDA).ipynb`) as well as music classification, where she implemented Naive Bayes classifiers (`bayes classification v3.ipynb`). Within the writeup, she wrote the sections on genre and composer classification (Sections 2.2, 3.2, and 4.2). Finally, for the presentation, she created the slides on classification.
- Hannah contributed to writing the Introduction, Model Description and Results Analysis for Music Composition sections. She participated in office hours and group meetings to discuss code and project design. She also helped create the presentation slides for the introduction and music composition metrics.

7 APPENDIX

7.1 Genre Classification

Table 4: Train-Test Split for Baroque Pieces in Genre Classification

Musical Piece		Composer	Train/Test Set
Concerto in C major Op. 7 No. 5	Part I	Tomaso Albinoni	Train
	Part II		Train
	Part III		Train
Concerto in C major, Op. 9 No. 9	Part I		Train
	Part II		Train
	Part III		Train
Sonata No. 3 in C major	-	Johann Sebastian Bach	Train
Prelude and Fugue in C major	-		Train
Violin Sonata No. 1 in g minor	Part I		Test
	Part II		Train
	Part III		Train
	Part IV		Test
Partita for Violin No. 1 in b minor	Part I		Train
	Part II		Test
	Part III		Train
	Part IV		Train
	Part V		Test
	Part VI		Test
	Part VII		Test
	Part VIII		Test
Violin Sonata No.2 in a minor	Part I		Train
	Part II		Train
	Part III		Train
	Part IV		Test
Violin Concerto in a minor	Part I		Test
	Part II		Test
	Part III		Train
Triple Concerto in a minor	Part I		Test
	Part II		Test
	Part III		Train
Violin Sonata in C major, Op.5 No.3	-	Arcangelo Corelli	Train
Concerto Grosso in a minor, Op. 6 No. 4,	Part I	George Frideric Handel	Train
	Part II		Test
	Part III		Test
	Part IV		Train
Violin Concerto in a minor		Antonio Vivaldi	Test

Table 5: Train-Test Split for Romantic Pieces in Genre Classification

Musical Piece		**Composer**	**Train/Test Set**
Capriccio No. 2 in b minor	-	Niccolò Paganini	Train
Capriccio No. 5 in a minor	-		Test
Capriccio No. 6 in g minor	-		Train
Capriccio No. 7 in a minor	-		Test
Capriccio No. 10 in g minor	-		Train
Capriccio No. 11 in C major	-		Test
Capriccio No. 16 in g minor	-		Train
Capriccio No. 18 in C major	-		Test
Capriccio No. 24 in a minor	-		Train
Étude Op. 10, No. 2 in a minor		Frédéric Chopin	Train
Mazurka in a minor, B. 134			Train
Mazurka in a minor, B. 140			Train
Mazurka in C major, B. 82			Train
Mazurkas, Op. 7	Part II: a minor		Test
	Part V: C major		Test
Mazurkas, Op. 17	Part IV: a minor		Train
Mazurkas, Op. 24	Part II: C major		Train
Mazurkas, Op. 33	Part III: C major		Test
	Part IV: b minor		Train
Mazurkas, Op. 56	Part II: C major		Test
Mazurkas, Op. 59	Part I: a minor		Train
Mazurkas, Op. 67	Part III: C major		Train
	Part IV: a minor		Test
Mazurkas, Op. 68	Part I: C major		Train
	Part II: a minor		Test
Waltz in B minor, Op. 69, No. 2	-		Test
Old French Song, Op. 39, No. 16	-	Pyotr Ilyich Tchaikovsky	Train
Trio for violin, cello and piano in a minor, Op. 50	Part I		Test
	Part II		Test
Sonata for violin and piano in a minor, Op. 105	-	Robert Schuman	Train
String quartets Op. 41	-		Train

7.2 Composer Classification

Table 6: Train-Test Split for Pieces Composed by Chopin in Composer Classification

Musical Piece		**Train/Test Set**
Étude Op. 10, No. 2 in a minor		Test
Mazurka in a minor, B. 134		Test
Mazurka in a minor, B. 140		Train
Mazurka in C major, B. 82		Train
Mazurkas, Op. 7	Part II: a minor	Test
	Part V: C major	Train
Mazurkas, Op. 17	Part IV: a minor	Train
Mazurkas, Op. 24	Part II: C major	Test
Mazurkas, Op. 33	Part III: C major	Train
	Part IV: b minor	Test
Mazurkas, Op. 56	Part II: C major	Train
Mazurkas, Op. 59	Part I: a minor	Train
Mazurkas, Op. 67	Part III: C major	Train
	Part IV: a minor	Test
Mazurkas, Op. 68	Part I: C major	Train
	Part II: a minor	Test
Waltz in B minor, Op. 69, No. 2	-	Train

Table 7: Train-Test Split for Pieces Composed by Paganini in Composer Classification

Musical Piece		**Train/Test Set**
Sonata No. 3 in C major	-	Test
Prelude and Fugue in C major	-	Test
Violin Sonata No. 1 in g minor	Part I	Test
	Part II	Train
	Part III	Train
	Part IV	Test
Partita for Violin No. 1 in b minor	Part I	Train
	Part II	Test
	Part III	Train
	Part IV	Train
	Part V	Train
	Part VI	Train
	Part VII	Train
	Part VIII	Train
Violin Sonata No.2 in a minor	Part I	Train
	Part II	Test
	Part III	Test
	Part IV	Test
Violin Concerto in a minor	Part I	Train
	Part II	Train
	Part III	Train
Triple Concerto in a minor	Part I	Test
	Part II	Train
	Part III	Test

Table 8: Train-Test Split for Pieces Composed by Bach in Composer Classification

Musical Piece		**Train/Test Set**
Sonata No. 3 in C major	-	Test
Prelude and Fugue in C major	-	Test
Violin Sonata No. 1 in g minor	Part I	Test
	Part II	Train
	Part III	Train
	Part IV	Test
Partita for Violin No. 1 in b minor	Part I	Train
	Part II	Test
	Part III	Train
	Part IV	Train
	Part V	Train
	Part VI	Train
	Part VII	Train
	Part VIII	Train
Violin Sonata No.2 in a minor	Part I	Train
	Part II	Test
	Part III	Test
	Part IV	Test
Violin Concerto in a minor	Part I	Train
	Part II	Train
	Part III	Train
Triple Concerto in a minor	Part I	Test
	Part II	Train
	Part III	Test