Reveal Hidden Information in the Music Scores: Composer Attribution

CS 229 Final Report

Fang-Chieh Chou,[*] Yi-Hong Kuo,[†] and Hsiang-Yu Yang[‡]

(Dated: 12/13/2013)

In this project, machine learning classification methods were applied to a music attribution challenge related to Josquin de Prez, a famous composer of the Renaissance period with many works misattributed to him. To solve the attribution problem, we trained classifiers using two kinds of music scores: scores verified to be composed by Josquin and scores composed by other contemporary musicians. These scores were retrieved from the Stanford Josquin Research Project. Features were extracted from the scores with knowledge of music theory and techniques of text mining. The optimized classifiers accurately distinguish whether a music score is composed by Josquin with a precision of 95% and a recall of 90%. An unsecure set of music works was analyzed by the optimized classifiers. From the successful results, our feature extraction and classification scheme may find useful applications in more general composer attribution problems.

I. INTRODUCTION

Just like the characteristic style of books can be used to attribute them to their author, music scores contain rich information about their composers. With modern machine learning (ML) methods, it is possible to extract hidden information from the music scores for attribution [1, 2]. One interesting challenge is related to Josquin des Prez (1450 - 1521), one of the most famous composers of the Renaissance period. Due to his immense reputation, many works were misattributed to him [3]. Among all the works attributed to Josquin, Only 31% has been verified, whereas for the rest the authenticity is still highly disputed. In this project, we used standard supervised classification ML methods to address this challenge.

II. DATA

Digitized music scores in the Humdrum format were obtained from the Josquin Research Project (JRP) [4]. Some of the works are collections of a few music scores, and we simply treated each score as an independent work. The scores were then labeled according to their composers: "Secure Josquin" for verified Josquin's work, "Unsecure Josquin" for doubtful Josquin's work, "Ockeghem" for works of Johannes Ockeghem, who was active 40 years earlier than Josquin, and "Others" for the works of composers contemporary with Josquin. The number of works in each category is listed in Table I. Works with more than one possible composers were removed from the training set to prevent dataset contamination.

Music21 [5], a Python-based music processing package, is utilized to analyze the Humdrum files.

Table I: The music score dataset

Data set	Number of Works
Secure Josquin	130
Ockeghem	92
Others	329
Unsecure Josquin	288

III. METHODOLOGY

For a given input music score, our goal is to confidently determine whether it is composed by Josquin. We used the "secure Josquin", "Ockeghem", and "Others" dataset to train the ML classifiers and estimate the error of our methods. The overall process works as follows (Figure 1): First, we extracted features with high discriminative power based on the idea of note-transitions and counterpoints from music theory. Second, the two different features were reduced in dimensions and combined into one set of composite features. The performance of the classifiers was evaluated using the stratified random sampling cross-validation scheme. Finally the optimized classifiers were applied to the unsecure Josquin dataset to determine the authenticity of those doubtful works. The details of each step outlined above are discussed below.

A. Feature Extraction

1. *Pitch-duration*

We first constructed a simple pitch-duration feature as a baseline model. Based on the idea of the bag-of-words model in text mining, a music score can be represented as a collection of its notes. The relations between notes, such as the order of notes, are ignored in this representation. The pitch-duration feature of a music work is a vector whose elements are the counts of different types of

[*] Electronic address: fcchou@stanford.edu
[†] Electronic address: stevekuo@stanford.edu
[‡] Electronic address: yanghy@stanford.edu

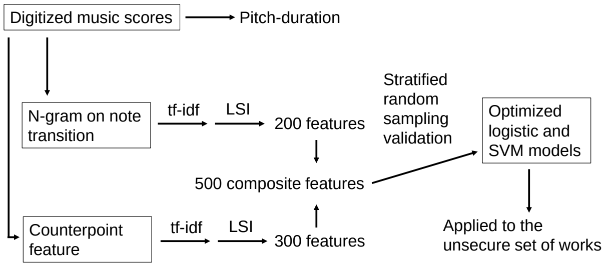
Digitized music scores
Pitch-duration
N-gram on note transition
tf-idf
LSI
200 features
Stratified random sampling validation
Optimized logistic and SVM models
500 composite features
Counterpoint feature
tf-idf
LSI
300 features
Applied to the unsecure set of works

Figure 1: The overall classification pipeline.

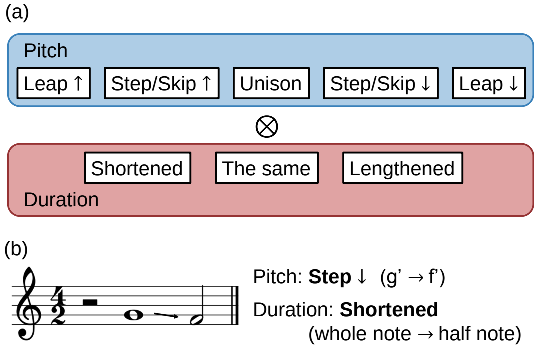

Figure 2: (a) 15 classes used as elements in the n-gram model. The changes in pitch are grouped into 5 categories. Intervals greater than a major third are identified as leaps, while intervals between a minor second and a major third are considered as steps/skips. The changes in duration are intuitively grouped into 3 categories. The direct product of pitch and duration classes gives 15 different transition classes. (b) An example showing how a transition class is assigned.

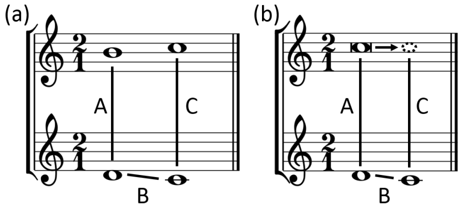

Figure 3: The counterpoint feature. (a) The notes in the minimum set of four notes are of equal length. The counterpoint tuple is (6, -2, 8) in this case. (b)The notes have different lengths.

notes appeared in the work. Each note is characterized by its pitch, which is represented by the MIDI number, and its time duration. As discussed in the Results section, this pitch-duration feature performs poorly. This result necessitates the development of more sophisticated features that capture the relationships between notes.

2. *N-gram on Note Transition*

To characterize the progression of notes in a single voice, we constructed a note transition feature based on the differences in pitch and duration between adjacent notes. Compared to the pitch-duration feature described above, the note transition is advantageous as it only uses the differences between notes. On the other hand, the pitch-duration feature is associated with the absolute pitches and lengths of individual notes, which vary dramatically across music works with different key and time signatures. In other words, the note transition can be considered as a standardization over key and time signatures.

We further simplified the feature set by categorizing the changes in pitch into 5 classes and the changes in duration into 3 classes. Therefore, all the possible note transitions can be categorized into 15 classes (Figure 2). The simplification reduces the dimension of the feature set while maintaining the essential characteristics.

The note transition feature described above only captures note correlation at the nearest-neighbor level. To capture the long range correlation between notes, i.e., the melody, we applied the n-gram method from text mining to the note transition feature. The n-gram method counts the number of occurrences of contiguous sequences of n transitions. In our case, the n-gram method leads to a feature dimension of 15^n. 2- and 3-gram were used for our final feature set.

3. *Counterpoint*

While the note transition feature characterizes the horizontal relationship within a voice, the vertical relationship between different voices, which is critical for polyphonic music, is not included. To capture this important feature, we utilized the concept of counterpoint from music theory. Consider a simple two-voice music shown in Figure 3(a). The relationship between the notes can be represented by their pitch differences, or intervals, labeled as A, B, and C in the figure. In cases where the notes have unequal lengths (Figure 3(b)), we virtually repeat a note to obtain the corresponding A, B, and C. Such scheme was applied to each pair of two voices in a score. The counterpoint feature is defined as the numbers of occurrences of all existing counterpoint tuples (A, B, C).

There are several ways to describe an interval. For example, the interval A in Figure 3(a) can be represented by 9 as expressed by the MIDI number difference, or Major 6th as the conventional name in music theory. In the project, the diatonic number (“6” in the example) of an interval is adopted. For the horizontal interval B, the direction of progression is also taken into account.

B. Feature Processing

1. Feature Weighting and Normalization

We applied the term frequency–inverse document frequency (tf-idf) scheme commonly used for text mining to process our features [6, 7]. Under this scheme, each term, which is an n-gram or a counterpoint tuple in our case, is re-weighted according to its importance. More specifically, the frequency of a term is adjusted to be the product of its tf and idf, where tf is the number of occurrences of a term in the score and idf the logarithm of the inverse ratio of the number of works containing the term to the total number of works. To avoid division-by-zero, Laplace smoothing is used in the calculation of idf.

The feature vectors for each work were then normalized to have unit L^2 norms. This normalization ensures the features of music scores with different lengths are comparable.

2. Feature Reduction and Combination

Another necessary feature processing is to reduce the dimension of our feature sets. Both the note transition and counterpoint feature vectors have thousands of components and those high dimensional feature sets lead to significant overfitting. The situation is even worse if we naively combine the two features as we roughly doubles the dimension of the feature space. Here the latent semantic indexing (LSI), a dimension reduction method widely used in texting mining, was applied. With LSI, we reduced the numbers of components of the note transition and counterpoint feature set to 200 and 300, respectively. These two reduced feature sets were then combined into one composite, 500-component set that we used for the classification.

C. Classifiers and Cross Validations

In this project we focused on two ML classification approaches: logistic regression and support vector machines (SVMs). Other classification methods such as naive bayes performed much worse, and therefore are not included in the discussion.

For cross-validation, we used the stratified random sampling method. This method has two advantages over regular k-fold cross-validation. First, the random sampling method allows arbitrary numbers of iterations, while the k-fold method allows only k iterations. Second, stratified sampling guarantees that the ratios of positive example in the generated test set and training set are the same as that of the whole dataset; the regular k-fold method does not guarantee this property and therefore leads to sampling error. This sampling method is important in reducing the variance of the computed validation statistics as our training data, in which only 28% of the samples is positive (Table I), is unbalanced and relatively small.

Typically test error, or accuracy equivalently, is used to evaluate the model. For this project, however, accuracy is inappropriate because of the imbalance of the dataset. For instance, even a classifier that assigns all the works as non-Josquin can achieve a high accuracy of 72% as only 28% of training data are positive. Instead, we focused on optimizing both the precision and recall of the classifiers. F_1 score, which is the harmonic mean of precision and recall, was used as the main validation statistic we aimed to maximize.

Scikit-learn [8], a Python ML package, was used to implement all the classifications, cross validations and evaluations discussed above.

IV. RESULTS AND DISCUSSION

We first trained the classifiers using "Ockeghem" and "Others" as negative samples. The accuracy, precision, recall, and F_1 score obtained by cross-validation of each ML algorithm-feature combination are summarized in the top half of Table II. Overall the SVM with the composite feature achieves the best performance, with a strong F_1 score of 91%. Unsurprisingly, the pitch-duration feature performs significantly worse than the composite feature. Besides, the F_1 score cannot be further improved by combining all three features together (result not shown). These results highlight the fact that the most important factor in distinguishing between composer styles is the relationship between notes.

It is worth noting that while using pitch-duration features leads to low F_1 scores, which indicates low precisions and low recalls, medium-level accuracy of 75 % is still achieved. This result is in agreement with our argument that the accuracy is not a proper statistic in evaluating our models. Compared with the classification scheme which simply assigns all works as non-Josquin, models with the pitch-duration feature perform only slightly better.

We visualized the separation power of the composite feature by projecting the validation data onto a plane normal to the decision boundary (Figure 4) in one of the cross validation iteration. We also plotted the precision-recall curves (Figure 5) for both classifiers. For the SVM, Platt's scaling method was used to obtain the prediction probability [9]. Both the logistic regression and the SVM gave areas under curve of 0.97 with 1.0 being the perfect case. We also plotted the precision and recall as functions of the probability threshold for classification. From the plot, the optimal threshold for the SVM is roughly at 0.5, which is the default threshold. On the other hand, the optimal threshold of logistic regression is at 0.35. This plot implies that a better performance can be achieved by adjusting the threshold. One application of those plots is to adjust the classifier to achieve the desired performance. For example, if high precision, say 95%, is prefered, we

Table II: Cross-validation performances of the classifiers.

	Classifier	Accuracy (%)	Precision (%)	Recall (%)	F_1 Score (%)
Pitch-duration(Baseline)	Logistic	75.8	59.1	52.2	54.6
	SVM	74.9	55.4	64.4	59.2
Composite	Logistic	95.0	94.2	88.2	90.2
	SVM	95.3	94.1	89.5	91.3
Contemporary composers	Logistic	94.7	96.2	88.8	92.1
	SVM	94.4	94.1	90.0	91.8
Ockeghem	Logistic	97.2	97.5	97.8	97.6
	SVM	97.7	97.9	98.3	98.0

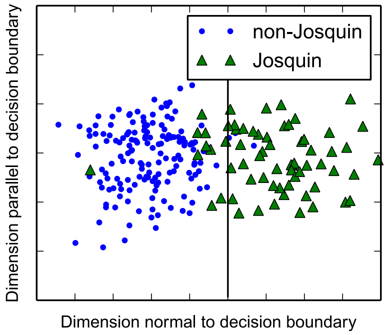

Figure 4: Visualization of the classification power of the composite features.

Table III: Prediction on the unsecure Josquin dataset

Model	Positive	Negative
Logistic	139	148
SVM	141	146

can set the probability threshold at 0.64 for the SVM classifier to fulfill that requirement.

So far we have been discussing the classifiers trained by the entire dataset. However, Johannes Ockeghem is one generation earlier than Josquin. To see the effect of composer era, we trained and tested classifiers using the secure Josquin set against either only the Ockeghem set or only the contemporary composers set. The results are summarized in the bottom half of Table II. We found that using just the Ockeghem dataset improve the performance significantly, while using only other contemporary composers does not. Indeed, our classifiers does a better job in separating works of Josquin and Ockeghem, but including Ockeghem into our training set does not deteriorate the classifier performance either.

Finally, we applied the optimized classifiers to the un-

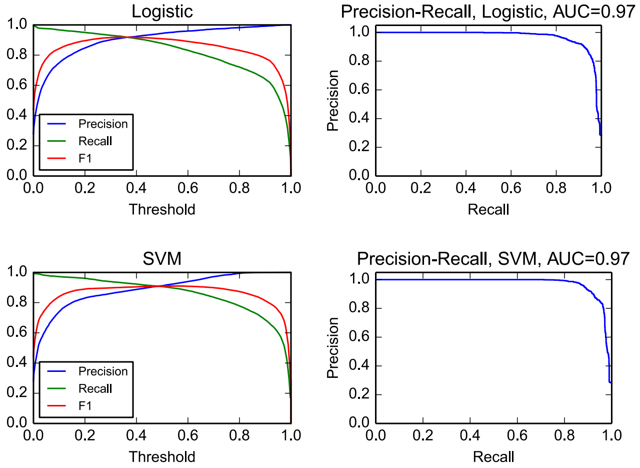

Figure 5: Left:Precision, recall and F_1 score as functions of threshold probability. Right: Precision-recall curves.

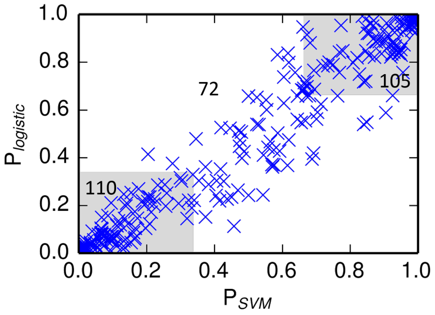

Figure 6: Scatter plot of $P_{logistic}$ against P_{SVM}, where $P_{logistic}$ and P_{SVM} are the probabilities of being authentic determined by the logistic regression and the SVM, respectively. Data points in the grey region at the top-right (bottom-left) corner are works highly-likely (highly-unlikely) to be composed by Josquin. The number of works of each region is indicated in the figure.

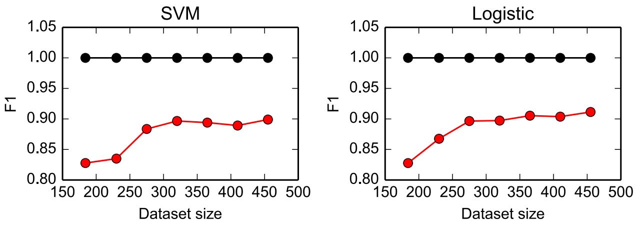

Figure 7: Learning curves of the SVM and the logistic models with composite features

secure dataset to obtain the probabilities of being authentic (Table III). To investigate the consistency between the SVM and the logistic regression, we plotted the probabilities obtained from the SVM against that from the logistic regression (Figure 6). The predictions are highly correlated between the two classifiers. This result supports the robustness of our ML scheme. Moreover, the majority of the data points are clustered near the top-right and bottom-left corners, indicating they are either highly-likely to be composed by Josquin or highly-unlikely to be composed by Josquin. Cases in the middle (what are in a sense the "uncertain" cases) only accounts for 25 % of the data.

While the classifiers have been carefully optimized, they are still overfitted as suggested by the learning curves (Figure 7). This result indicates that obtaining more training data, such as more music works from other composers contemporary with Josquin, may be beneficial.

V. CONCLUSION

We have successfully applied ML techniques to solve the composer attribution challenge of Josquin. With features extracted base on music theory and text mining, we achieved highly-confident classfications as validated by cross validation. Furthermore, the optimized classifiers gave consistent predictions on the unsecured set.

Our work demonstrates that a comprehensive feature extraction scheme that captures the relationships between the notes in music scores, both horizontally as the music flows over time and vertically between different voices, is the key factor in extracting the hidden information. While our feature extraction scheme has only been tested in this Josquin challenge, we believe that the same scheme can be applied to more general music style attribution applications.

Acknowledgments

The authors acknowledge Jesse Rodin and Craig Sapp for proposing the project, providing the digitized music scores and useful discussions.

[1] L. Mearns, D. Tidhar, and S. Dixon, "Characterisation of composer style using high-level musical features," in *Proceedings of 3rd International Workshop on Machine Learning and Music*, ser. MML '10, 2010, pp. 37–40.

[2] M. Hontanilla, C. Perez-Sancho, and J. Inesta, "Modeling musical style with language models for composer recognition," in *Pattern Recognition and Image Analysis*, ser. Lecture Notes in Computer Science, 2013, vol. 7887, pp. 740–748.

[3] J. Rodin, "Josquin and epistemology."

[4] "The josquin research project." [Online]. Available: http://josquin.stanford.edu/

[5] "music21: A toolkit for computer-aided musicology." [Online]. Available: http://web.mit.edu/music21/

[6] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," in *Information Processing and Management*, 1988, pp. 513–523.

[7] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Machine Learning: ECML-98*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, vol. 1398, pp. 137–142.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[9] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers.* MIT Press, 1999, pp. 61–74.